Advant® OCS
with Master software

AdvaBuild® Diagram Builder

User’s Guide

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

ABB Master, Advant, AdvaCommand, AdvaInform, AdvaBuild, and AdvaControl are registered trademarks of ABB Asea Brown Boveri Ltd. Switzerland. MOD 300 is a trademark of ABB Process Automation Inc.

HP and all HP-products mentioned in this publication are registered trademarks of Hewlett Packard Company

ORACLE, PL/SQL; and SQL*Plus are registered trademarks of Oracle Corporation

UNIX is a registered trademark of AT & T Corporation

AutoCAD and DXF are a registered trademark of Autodesk Inc.

FrameMaker is a trademark of Frame Technology Corp.

IBM-PC is a trademark of IBM Corporation

OSF/Motif is a trademark of The Open Software Foundation, Inc.

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Industrial Systems AB. ABB Industrial Systems AB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Industrial Systems AB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Industrial Systems AB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Industrial Systems AB’s written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

CE MARKING

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Copyright © ABB Industrial Systems AB 1997

3BSE001264/D

Template: 3BSE001286/E

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 General Information 1-1
1.2 Equipment Requirements 1-2
1.3 Manual Organization 1-2
1.4 Related Documentation 1-2
1.5 Terminology 1-5
  1.5.1 Diagram/Drawing Type Definitions 1-5

Chapter 2 - Installation

2.1 Installation of (Electrical) Diagram Builder 2-1
2.2 Installation of Diagram Builder Related Libraries on File System 2-1
  2.2.1 Structure of Delivered Symbol Libraries 2-1
  2.2.2 Structure of Delivered Format Libraries 2-4
  2.2.3 Installation Procedure 2-5

Chapter 3 - Configuration and Application Building

3.1 Design Considerations 3-1
  3.1.1 Directory Structure Overview 3-1
  3.1.2 Structure and Handling of Initial Drawing Files 3-2
  3.1.3 Structure and Handling of Configuration Files 3-3
  3.1.4 Structure of Format Libraries 3-5
  3.1.5 Handling of Logos in Format Library 3-5
  3.1.6 Structure of Project Symbol Libraries 3-6
  3.1.7 Structure of Plotter Settings Directories 3-7
  3.1.8 Stand-Alone Environment 3-8
3.2 Capacity and Performance 3-8
3.3 Application Start-up 3-8
3.4 Configuration Tutorial 3-9
  3.4.1 Installing DIB/(E)DIB 3-9
  3.4.2 Installing Symbol and Format Libraries in Project 3-10
    3.4.2.1 Setup for configuration 3-10
    3.4.2.2 Installing symbol and format libraries in project 3-10
    3.4.2.3 Generating slide menus for symbol sublibrary 3-13
  3.4.3 Adaptation of title block mapping in Project 3-14
    3.4.3.1 Setup for configuration 3-14
    3.4.3.2 Configuration of file dib.sql 3-14
    3.4.3.3 Configuration of file dib.dmi 3-14

CONTENTS (continued)

3.5 Configuration Procedures 3-20
3.5.1 Symbol and Format Library handling functions 3-20
3.5.1.1 Creating Symbol Libraries 3-20
3.5.1.2 Updating Symbol Libraries 3-22
3.5.1.3 Creating Format Libraries 3-23
3.5.1.4 Updating Project-specific Format Libraries 3-25
3.5.2 Configuring (Electrical) Diagram Builder 3-25
3.5.2.1 Configuring data base Retrieval File 3-26
3.5.2.2 Configuring Object Type Specific Setup Files 3-29
3.5.2.3 Configuring Title Block Mapping File 3-32
3.5.3 Configuring Plotter Settings 3-32
3.5.4 Configuring AutoCAD 3-36
3.5.4.1 Configuring Operating Parameters 3-37
3.5.4.2 Configuring Plotters 3-38
3.5.4.3 Configuring Video Display 3-39
3.6 Configuration Menus 3-40

Chapter 4 - Operation

4.1 Product Operation 4-1
4.2 Operation Overview 4-1
4.2.1 General Mechanisms 4-1
4.2.2 Privileges 4-3
4.2.3 Locking Mechanisms 4-3
4.2.4 Failure Mechanisms 4-5
4.3 Operation Tutorial 4-8
4.3.1 Building Diagrams 4-8
4.3.1.1 General Aspects 4-8
4.3.1.2 STB/TEB Operations 4-8
4.3.1.3 Common DIB/EDIB Operations 4-10
4.3.1.4 DIB Operations 4-12
4.3.1.5 EDIB Circuit Operations 4-12
4.3.1.6 EDIB Layout Operations 4-15
4.3.2 Building Symbol Blocks 4-17
4.3.2.1 General Aspects 4-17
4.3.2.2 STB/TEB Operations 4-17
4.3.2.3 DIB operations 4-17
4.3.3 Building Composed Symbol Blocks 4-20
4.3.3.1 General Aspects 4-20
4.3.3.2 DIB operations 4-20

CONTENTS (continued)

4.4 Operation Instructions ... 4-20
  4.4.1 Configuring AutoCAD ... 4-20
  4.4.2 Configuring Diagram Builder ... 4-21
  4.4.3 STB/TEB Operations ... 4-21
    4.4.3.1 Inserting Diagram Objects in AdvaBuild ... 4-21
    4.4.3.2 Editing Diagram Object Attributes ... 4-23
    4.4.3.3 Opening Diagram Objects with Diagram View ... 4-25
    4.4.3.4 Deleting Diagram Objects ... 4-25
    4.4.3.5 Copying Diagram Objects with Structure Builder ... 4-26
    4.4.3.6 Copying Diagram Objects with Template Builder ... 4-26
    4.4.3.7 Renaming Diagram Objects ... 4-26
    4.4.3.8 Printing Diagrams Directly ... 4-27
    4.4.3.9 Printing Diagrams with Plant Documentation Builder .. 4-28
  4.4.4 DIB Operations ... 4-29
    4.4.4.1 General Operations ... 4-29
    4.4.4.2 Editing Drawings ... 4-29
    4.4.4.3 Editing P&I Diagrams ... 4-29
    4.4.4.4 Editing Hookup Diagrams ... 4-29
    4.4.4.5 Editing Layout Diagrams ... 4-29
    4.4.4.6 Editing Circuit Diagrams ... 4-29
    4.4.4.7 Handling of Item Designations in Circuit Diagrams ... 4-29
4.5 Operation Menus ... 4-33

Chapter 5 - Maintenance

5.1 EXPORT/IMPORT of Existing Drawings ... 5-1
  5.1.1 How to Reuse Existing Drawings ... 5-1
  5.1.2 How to Import a Drawing from a PC ... 5-2
5.2 Trouble Shooting ... 5-2
  5.2.1 (Electrical) Diagram Builder Starts Up With the AutoCAD Menu. ... 5-2
  5.2.2 AutoCAD Cannot Find/access Existing Files During Insert. ... 5-3
  5.2.3 FILLMODE is set to ON in AutoCAD ... 5-3
  5.2.4 Printing Diagrams from the Plant Documentation Builder or the Structure Builder ... 5-3
  5.2.5 (Electrical) Diagram Builder cannot find liblist.txt files ... 5-3
  5.2.6 Multiple Configuration Points within a Drawing ... 5-4
  5.2.7 Title Block Update does not Work ... 5-5

CONTENTS (continued)

5.2.8 Display of Colors when Working in Parallel with DIB and DYB or DOB 5-5
5.2.9 Creating Slides for Symbol Libraries 5-5
5.2.10 Reserving an AutoCAD Licence 5-5
5.3 Backup/Restore 5-6

Appendix A - Guideline for Building Formats

A.1 General rules A-1
A.2 Standard Title Block Attributes A-3
A.3 Optional Title Block Attributes A-4
A.4 Revision Block Attributes A-4
A.5 Logo A-5
A.6 Grid reference system A-5
A.7 Sheet layout A-6
A.8 Title/Revision Block Layout A-10
A.9 Scaling, rotation and offset for printout A-12
A.10 Naming Conventions A-13

Appendix B - Structuring and Naming Conventions for Symbol Libraries

B.1 Introduction B-1
  B.1.1 Summary B-1
  B.1.2 Background B-1
  B.1.3 Dependencies B-1
B.2 General Naming Rules B-2
  B.2.1 General Symbol Name Conventions B-2
  B.2.2 Symbol Classification Code B-4
  B.2.3 Device Classification Code B-5
  B.2.4 Supplier/Manufacturer Code B-6
  B.2.5 Symbol Orientation Code B-6
  B.2.6 Refinement of Device Classification Code Block B-7
  B.2.7 Symbol Numbering Concerning IEC 617/DIN B-9
B.3 Application specific Mapping of Symbols to Library (sub)directories B-12
  B.3.1 Circuit Diagram Application B-12
    B.3.1.1 Electrical and Instrumentation Symbols B-12
    B.3.1.2 Cross-reference Symbols B-13
    B.3.1.3 Miscellaneous Symbols B-13

CONTENTS (continued)

B.3.2 Layout Diagram Application (Arrangement) B-14
B.3.2.1 Symbols for Arrangement B-14
B.3.2.2 Miscellaneous B-15
B.3.2.3 Recommended Conventions for Not Yet Considered Layout Symbols B-15
B.3.2.4 Symbols for Application Specific Symbols B-15
B.3.3 Hookup Diagram Application (Assembly) B-16
B.3.3.1 Mechanical Symbols for Hookup B-16
B.3.3.2 Miscellaneous B-16
B.3.4 P&I Diagram Application B-16
B.3.4.1 P&I Diagram Symbols B-17
B.3.4.2 Single Line Symbols B-17
B.3.4.3 Miscellaneous B-18
B.3.5 Drawing Application B-18
B.3.5.1 Graphic Symbols B-18
B.3.5.2 Single Line Symbols (for Overview Diagrams) B-18
B.3.5.3 Block Symbols (for Overview Diagrams) B-18
B.3.5.4 Miscellaneous Symbols (for Overview Diagrams) B-19

Appendix C - Guideline for Building Symbols

C.1 Introduction C-1
C.1.1 Background C-1
C.1.2 General Conventions C-1
C.2 Circuit diagram symbols C-1
C.2.1 Reference to standards C-1
C.2.2 Graphical design principles C-1
C.2.3 Attribute conventions C-2
C.2.4 Connection Point and Reference Point Definitions C-5
C.2.5 Symbol Groups/Classes C-6
C.2.5.1 Mother symbols for devices C-7
C.2.5.2 Daughter symbols for device channels C-8
C.2.5.3 Cross-reference symbols for device channels C-9
C.2.5.4 Terminal symbols C-10
C.3 P&I Diagram/Drawing symbols C-10
C.3.1 Reference to standards C-10
C.3.2 Graphical design principles C-11
C.3.3 Attribute conventions C-12

CONTENTS (continued)

C.4 Layout symbols C-12
  C.4.1 Graphical design principles C-12
  C.4.2 Attribute conventions C-12
C.5 Hookup symbols C-14
  C.5.1 Graphical design principles C-14
  C.5.2 Attribute conventions C-14

Appendix D - Internal Variables and Environment Variables

D.1 Introduction D-1
D.2 Syntax Definition D-1
D.3 Semantic Definition D-3
D.4 Complex Examples D-4
  D.4.1 Checking for an Empty String D-4
  D.4.2 Defining Mappings D-4
  D.4.3 Comparing Values D-5

INDEX

Chapter 1 Introduction

1.1 General Information

This manual explains how to install, configure, operate, and maintain the AdvaBuild Diagram Builder. The Diagram Builder is one of two AdvaBuild tools used to create diagrams. The other tool is the AdvaBuild Document Builder software, a comprehensive workstation publishing software package offering both top-quality word processing and a simple drawing tool.

The **Diagram Builder** consists of:

- AutoCAD (as base system)—a multipurpose drawing tool with powerful CAD capabilities. You can create complex drawings featuring small quantities of text with Diagram Builder.
- DIB Application (as AutoCAD application)—corresponds to the product MasterLine from S&S Company and is used for handling general purpose diagrams and handling of symbol libraries
- Diagram Builder interface (for coupling to AdvaBuild Engineering Data base).

The Diagram Builder can be enhanced by adding the **Electrical Diagram Builder**, which is further subdivided into the applications:

- Electrical Diagram Builder Circuit Diagrams (for CI_DIA objects)
- Electrical Diagram Builder Layout Diagrams (for LA_DIA objects).

Functions for building new symbol blocks are included in all applications.

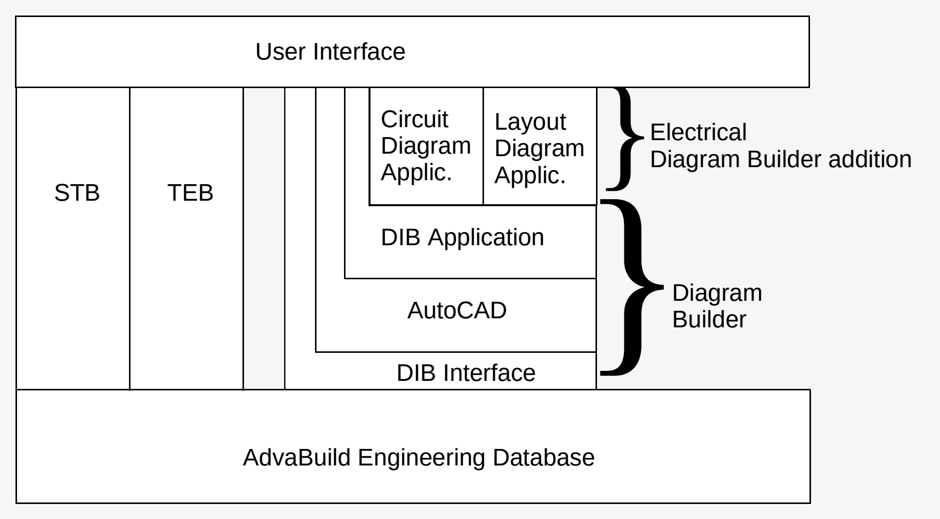


Figure 1-1. Diagram Builder integration in AS500ES

1.2 Equipment Requirements

You can install the AdvaBuild Diagram Builder on any Advant Station 500 Series configured as an Engineering Station (see *Advant Station 515/520 Hardware Reference Manual*).

1.3 Manual Organization

This user's manual is organized based on the structure outlined below.

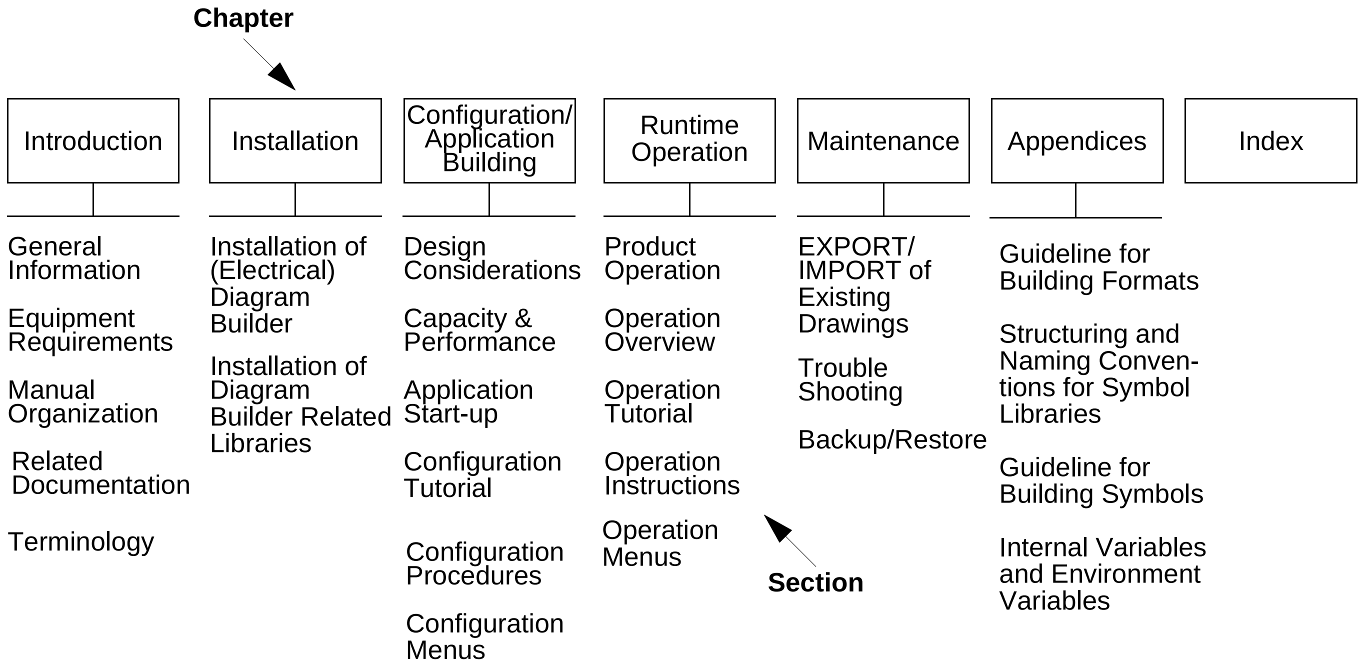


Figure 1-2. Organization of Diagram Builder User's Manual.

1.4 Related Documentation

This manual is part of AdvaBuild user documentation. The *AdvaBuild Basic User's Guide* is the primary manual in the AdvaBuild documentation series.

Standards related to the Diagram Builder and/or referenced within the manual are listed in the following table:

Table 1-1. Diagram Builder Standards

Ref.no.	Document Identity	Document Title
[I1]	CEI/IEC 1082	Preparation of documents in electrotechnology -1: General requirement -2: Function oriented diagrams
[I2]	IEC 617-1..13	Graphical symbols for diagrams
[I3]	IEC/SC3B(Sec)92	Standard character set to be used in electrotechnical drawing
[I4]	IEC/SC3B(Sec) 130	Addition to IEC 1082-1 "Additional rules for simplification techniques of item designations"
[I5]	IEC/SC3B (Sec)122	Structuring Principles and Reference Designations
[I6]	IEC 750 Edition 1983	Item designation in electrotechnology
[I7]	IEC 3B(Sec) 100	Preparation of documentation for plants, system and equipments
[S1]	ISO 3098	Technical Product Documentation -10: General requirements -11: CAD-lettering of Latin alphabet, ciphers and markers
[S2]	ISO 3511-1/2/3/4:	Process measurement control functions and instrumentation
[S3]	ISO 5457	Technical drawings - sizes and layout of drawing sheets
[S4]	ISO 7200	Technical drawings - Title blocks
[S5]	IEC 3(Sec)460 ISO TC145 N250	General principles for creation of symbols for use in documentation of technical products
[S6]	IEC/3A(Sec)307	Graphical symbols for diagrams, Part 1: General rules, general index
[S7]	ISO/DIS 10 628	Flow diagrams for process plants Part1: General rules
[S8]	ISO 8859-1...9	Information processing; 8-bit single-byte code graphic character sets
[S9]	ISO/IEC 10303-212	Electrotechnical plants, External Representation of Product Definition Data

Table 1-1. Diagram Builder Standards (Continued)

Ref.no.	Document Identity	Document Title
[S10]	ISO 1291	Fluid power systems and components - 1: Graphic symbols - 2: Circuit diagrams
[D1]	DIN 32 830	Graphical Symbols, Rules for creation of graphical symbols for use in technical documentation
[D2]	DIN 40 719	Diagrams, charts, tables T1: Definitions and classifications T2: Item designations T3: Recommendations for the preparation of circuit diagrams T1000: (Draft) Rules for computer-aided design of electrotechnical circuit diagrams
[D3]	DIN 6776 T1	Technische Zeichnungen, Beschriftung Schriftzeichen
[D4]	DIN V 6779	Structuring principles for technical products and technical product documentation -1: Basic rules and principles -2: Tables for identification letters
[D5]	DIN 19227	Control technology, Graphical symbols and identifying letters for process control engineering 1: Symbolic representation for functions 2: Representation of details
[D6]	DIN V 40900 T1000	Graphical symbols for diagrams; Rules for computer-aided construction of graphical symbols
[D7]	DIN 28004	Flowsheets and diagrams of process plants - 1: Definitions - 2: Drawing Instructions - 3: Graphical symbols
[A1]	Basic B6 (92-8)	ABB Corporate Standard: Drawing Rules
[A2]	B19 (93-05)	ABB Corporate Standard: Documentation of Deliveries Kind of Documents
[A3]	B20 (93-05)	ABB Corporate Standard: Documentation of Deliveries General Principles

1.5 Terminology

1.5.1 Diagram/Drawing Type Definitions

Circuit diagram
[I1]"A circuit diagram shall show the details of the implementation of a system, subsystem, installation, equipment, etc. without necessarily taking into account the physical sizes, shapes or locations and the constituent items."
A circuit diagram is handled in AdvaBuild by object type CI_DIA.

Arrangement drawing
[I1]"Assembly drawing simplified or supplemented to give information needed for a particular purpose. "
An arrangement drawing is handled in AdvaBuild by object type LA_DIA.

Assembly drawing
[I1]"Drawing representing the spatial position and shape of a group of assembled parts, normally to scale."
An assembly drawing is not yet handled in AdvaBuild by specific object type and specific functions can be handled as general drawing (object type DRAWING).

Hookup diagram
Special type of assembly drawing showing the assembly of measuring and actuating equipment for field instrumentation. A hookup diagram is handled in AdvaBuild by object type HOOKUP.

P&I diagram
[S7]"The piping and instrument diagram, based on the process flow diagram, represents the technical realization of a process by means of graphical symbols for equipment and piping together with graphical symbols for process measurement and control functions. "
A P & I diagram is handled in AdvaBuild by object type PI_DIA.

Overview diagram
[I1]" Relatively simple diagram, often using single line representation, showing the main interrelations or connections among the items within a system, sub-system, installation, part, equipment, software, etc."
An overview diagram is not yet handled in AdvaBuild by specific object type and specific functions can be handled as general drawing (object type DRAWING).

Block diagram
[I1]"Overview diagram using block symbols predominantly. "
[S7]"The block diagram represents a process plant in simple form with the aid of rectangular boxes including the relevant inscriptions, interconnected by flow lines."
A block diagram is regarded as a special form of an overview diagram.

Single line diagram
A single line diagram is a special form of an overview diagram using single line symbols.

Chapter 2 Installation

2.1 Installation of (Electrical) Diagram Builder

The Diagram Builder is an AdvaBuild option. The Electrical Diagram Builder is considered additional functionality, which requires the Diagram Builder.

The procedure for installing AdvaBuild Diagram Builder and Electrical Diagram Builder software is part of the AS500 ES Release Notes shipped with your Advant Station 500 Series workstation (configured as an Engineering Station).

2.2 Installation of Diagram Builder Related Libraries on File System

Installation of Diagram Builder related libraries provides you with:

- symbol libraries
- frame/format libraries.

You can copy the library components for use within projects. The procedure for those activities is described in Chapter 3, Configuration and Application Building.

2.2.1 Structure of Delivered Symbol Libraries

This section describes how AdvaBuild Diagram Builder symbol libraries are delivered and located within the standard file structure of AdvaBuild. Figure 2-1 shows an example structure.

Responsibility Level is separated related to the responsibility into the following three paths referenced by predefined environment variable:

```
${C_STA_ES_LIB_CSYM}/...    #Cad Symbol Libraries with AdvaBuild System responsibility
${C_ADA_ES_LIB_CSYM}/..     #Cad Symbol Libraries with BU/BA responsibility/adaptions
${C_CUS_ES_LIB_CSYM}/..     #Cad Symbol Libraries with Customer responsibility/adaptions
```

On this level a `.lis` file can also be available, which includes references to library installation script files. Those files are copied automatically into the project-specific DIB configuration directory.

Below are several Classification Levels. The number and use of levels is principle free. But the Diagram Builder libraries should follow the convention

.../<aspect>/<standard>/<language>/<symbol class>[/<subclass>]/<release>/<symbol>`.dwg`

Examples for <*aspect*> are:

- cis: for circuit diagram symbols
- las: for layout diagram symbols
- hos: for hookup diagram symbols
- pis: for P&I diagram symbols
- drs: for drawing symbols.

Examples for <*standard*> are:

- iec: for IEC standard
- din: for DIN standard
- bay: for Bayer standard
- abb: for ABB standard.

Examples for <*language*> are:

- en: for english
- se: for swedish
- ge: for german
- nol: no language dependencies.

Examples for <*symbolclass*> are:

- motors: for motor symbols in circuit diagrams
- plc: for PLC/DCS symbols in circuit diagrams
- pinstru: for instrumentation symbols in P&I diagrams.

NOTE

The namings for <*symbolclass*> should correspond to sublibrary namings described in appendix B.

The next level <*subclass*> is optional and should only be used if too many symbols would be included below <*symbolclass*> and only parts of them become relevant within a project.

This will become relevant for PLC/DCS symbols, where only the symbols for a certain product series may be relevant:

Examples for <*subclass*> below directory *plc* are:

- **`mas-s100`**: for Master S100 I/O
- **`adv-s600`**: for Advant S600 I/O

Examples for release may be:

- **`r1_2:`** for release R1.2
- **`r2_0-1`**: for release R2.0/1.

NOTE

Symbol names must be unique within the scope of a symbol class directory including all subclass directories.

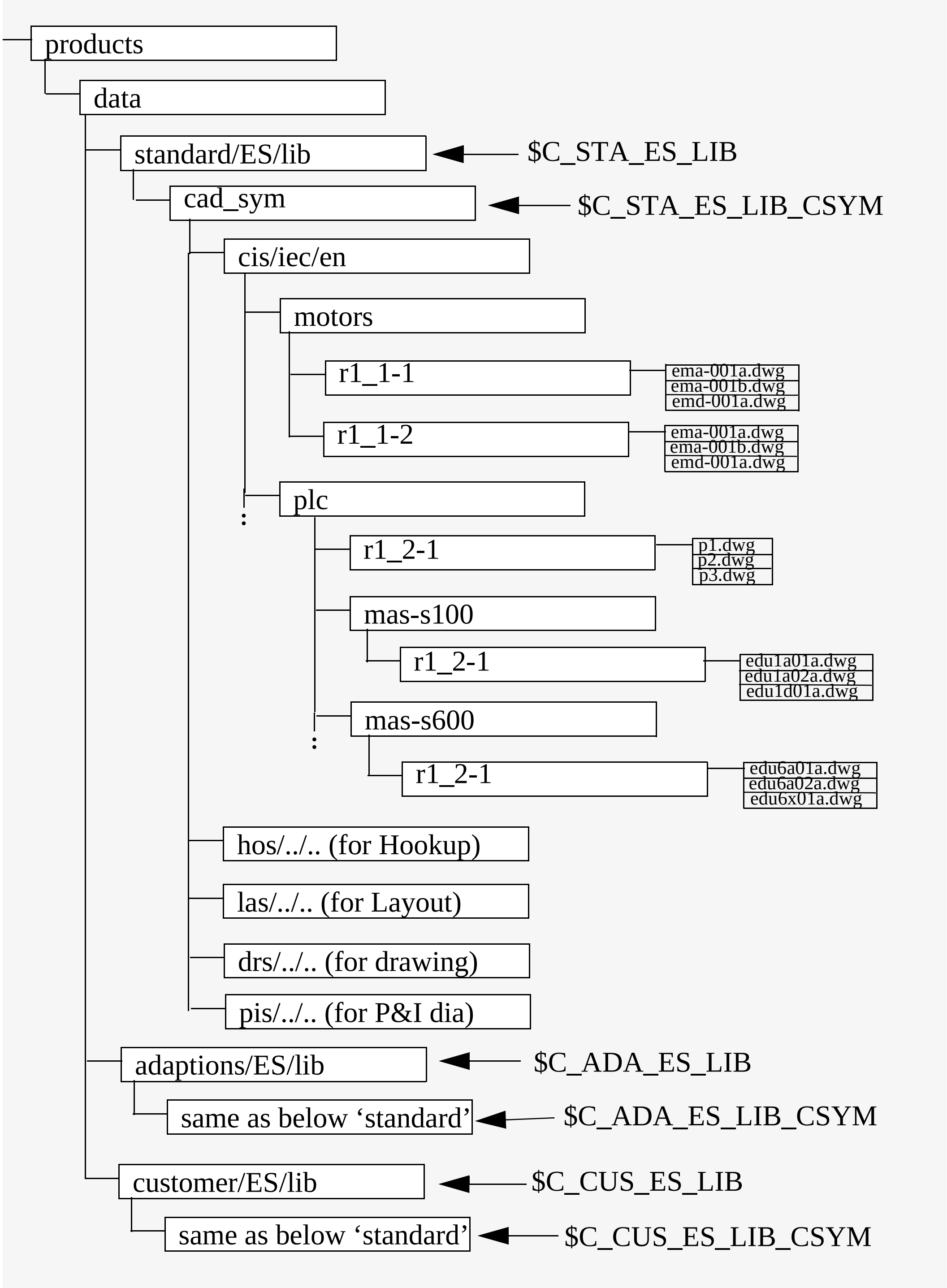


Figure 2-1. Example for a symbol library delivery structure

2.2.2 Structure of Delivered Format Libraries

This section describes how AdvaBuild Diagram Builder format libraries are delivered and located within the standard file structure of AS500ES.

Responsibility Level is separated into the following three paths:

```
${C_STA_ES_LIB_CFORM}/...    #Cad Format Libraries with AdvaBuild System responsibility
${C_ADA_ES_LIB_CFORM}/...    #Cad Format Libraries with BU/BA responsibility/adaptions
${C_CUS_ES_LIB_CFORM}/..     #Cad Format Libraries with Customer responsibility/adaption
```

A `.lis` file may also be available at this level, which includes references to library installation script files. Those files are copied automatically into the project-specific DIB configuration directory.

Below are several Classification Levels. The number and use of levels is principle free but the AdvaBuild Diagram Builder libraries should follow the convention:

.../<aspect>/<standard>/<language>/format/<release>/<formatname>.dwg
/logo<release>/<logoname>.dwg

Examples for <*aspect*> are:

- gef: for general diagram formats
- cif: for circuit diagram formats
- laf: for layout diagram formats
- hof: for hookup diagram format
- pif: for P&I diagram formats
- drf: for drawing formats.

Examples for <*standard*> are:

- iec: for IEC standard
- din: for DIN standard
- bay: for Bayer standard
- abb: for ABB standard.

Examples for <*language*> are:

- en: for english
- se: for swedish
- ge: for german
- nol: no language dependency.

Examples for <*formatname*> are:
aol, a1l, a2l, a3l, a3p, a4l, a4p

Examples for <*release*> may be:

- r1_2-1: for release R1.2/1
- r2_0-1: for release R2.0/1.

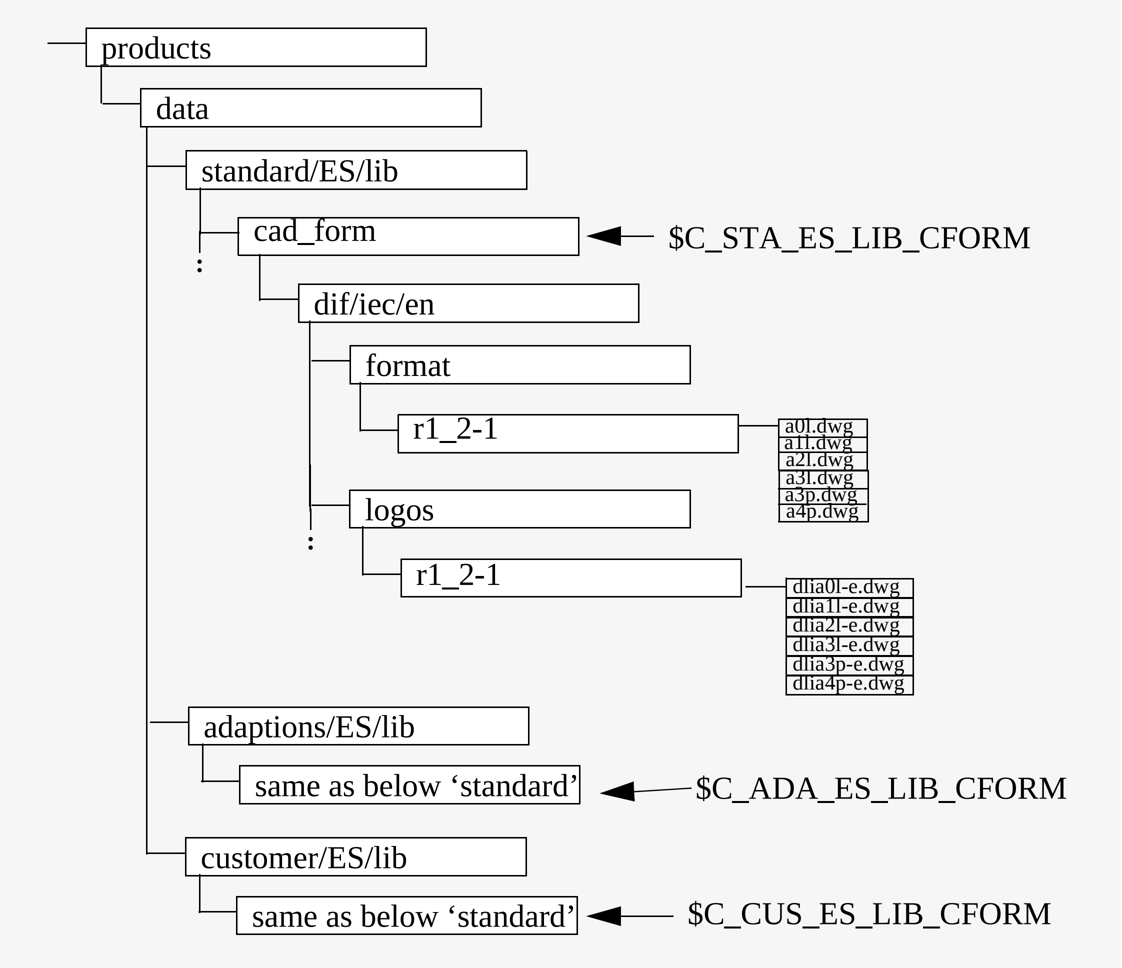


Figure 2-2. Example for a format/frame library delivery structure

2.2.3 Installation Procedure

The procedure for installing AdvaBuild Diagram Builder and Electrical Diagram Builder libraries is described in the *Advant Station Engineering Station Release Notes R2.1/0.*

Chapter 3 Configuration and Application Building

3.1 Design Considerations

3.1.1 Directory Structure Overview

AdvaBuild Diagram Builder's directory structure is organized into the following levels:

- System level for files that are part of the product
- Global level for overall project purposes
- Project level for single project purposes
- User level for special user purposes.

Each level contains part of an AdvaBuild directory structure subtree. Within AdvaBuild, directories are normally defined by environment variables; only these variables are used in the AdvaBuild Diagram Builder (E)DIB. In this chapter, an environment variable, for example, C_BIN, refers to the directory defined by $C_BIN.

System level directories are:

- C_BIN_CAPE, C_BIN_CAPE_BIN, C_BIN_CAPE_MSC, C_BIN_CAPE_SH.

Global level directories are:

- $C_CAPE_ALL/acad.

  This directory is available for historical purposes if the current installation was upgraded from prior releases (not in new installations). The directory and all its subdirectories will be removed in the next release.

  NOTE

  Please merge all files you want to copy to other directories.

- $C_CAPE_ALL_CFG_DIB/mouse and $C_CAPE_ALL_CFG_DIB/digit

  These directories are used for AutoCAD configuration files. The directory $C_CAPE_ALL_CFG_DIB/mouse contains the AutoCAD configuration that is used when working on X-Terminals. The directory $C_CAPE_ALL_CFG_DIB/digit is used when working on the console.

- C_CAPE_ALL_MSC_DIB

  This directory is used for overall configuration of the Diagram Builder. This includes:

  - Plotter/Printer configuration
  - Diagram Builder overall configuration/customization files

Project level directories are:

- $C_PROJ/acad.

  This directory is available for historical purposes if the current installation was upgraded from prior releases (not in new installations). The directory and all its subdirectories will be removed in the next release.

  NOTE

  Please merge all files you want to copy to other directories. The project which will be initially created in this release will not contain this directory.

- C_PROJ_LIB_CAD.

  This directory is used for project specific symbol libraries

- C_PROJ_TPL_CAD.

  This directory is used for project specific format libraries

- C_PROJ_MSC_DIB.

  This directory is used for project specific configuration of the Diagram Builder. This includes:

  - Application specific configuration of symbol and format library setup
  - Definition of title block update rules.

User level directories

- $C_USR/acad.

  This directory is available for historical purposes if the current installation was upgraded from prior releases (not in new installations). The directory and all its subdirectories will be removed in the next release.

  NOTE

  Please merge all files you want to copy to other directories. The project which will be initially created in this release will not contain this directory.

3.1.2 Structure and Handling of Initial Drawing Files

Initial drawing files are very similar to AutoCAD prototype drawings. The only difference is that the (Electrical)Diagram Builder copies initial drawing files to create new diagrams. (E)DIB does not use the AutoCAD feature, where one can start AutoCAD without specifying a working file and let AutoCAD do the job to assign an adequate prototype.

(E)DIB uses the following initial drawing files:

Table 3-1. Mapping of initial drawing to application

filename	application	filename	application
ci_dia_em.dwg	EDIB	pi_dia_ml.dwg	DIB, EDIB
la_dia_em.dwg	EDIB	hookup_ml.dwg	DIB, EDIB
ci_dia_ml.dwg	DIB	drawing_ml.dwg	DIB, EDIB
la_dia_ml.dwg	DIB		

Depending on which option is active, the initial drawing as assigned above is taken.

Within the initial drawing file various settings are defined. Settings are grouped into two parts:

- AutoCAD settings, such as:
  - system variables
  - predefined blocks
  - line types and text styles
  - menu file reference.
- AdvaBuild Diagram Builder settings in the Basic Configuration dialog box which can be reached via DIBCONFIG CONFIG from the (E)DIB screen menu.

3.1.3 Structure and Handling of Configuration Files

When the Diagram Builder is started on a diagram object, a start-up procedure is executed.

The start-up procedure reads a number of configuration files. There are configuration files on system level and on project level. For customization only the configuration files on project level are relevant.

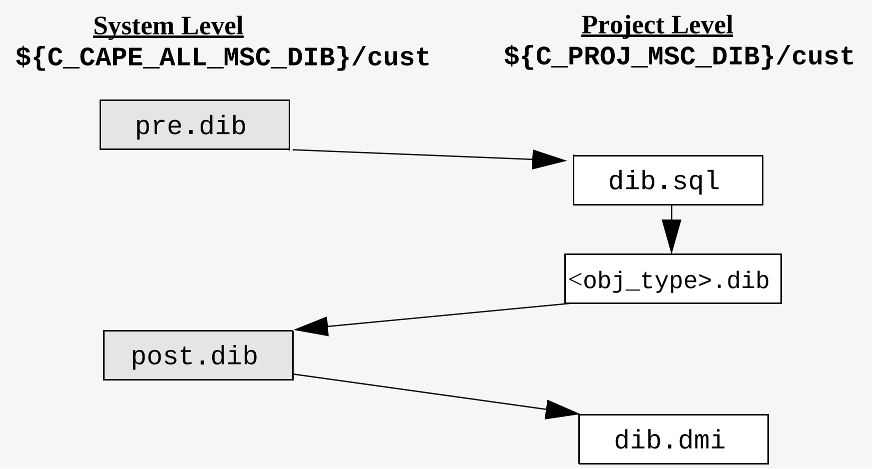


Figure 3-1. Scheduling of configuration files

Configuration files on system level are normally not visible to you and must not be changed. However, an error may occur when loading those files. In such cases the depicted error message points to some error within either pre.dib or post.dib. Normally such errors occur because of undefined environment variables and will not need to be corrected within the mentioned files.

For each file as counted above a corresponding lis-file will be created in case of errors.

CAUTION

> Do not make changes in the files pre.dib and post.dib even if a lis-file exists for these files, because some error was detected during load.

Short explanation of configuration files:

1. `pre.dib`

Some important preparations. Not relevant for configuration.

2. `dib.sql`

Here the selection of attributes from the data base is described relevant for title block update during open and print.

3. <*obj_type*>`.dib`

Here object and project specific configuration settings are described, such as:

- symbol library setup
- format library setup
- plot setup

Information related to object types is separated among the following files:

- `ci_dia.dib`
- `la_dia.dib`
- `pi_dia.dib`
- `hookup.dib`
- `drawing.dib`

4. `post.dib`

Some important post preparations. Not relevant for configuration.

5. `dib.dmi`

Here the mapping between attributes selected from the data base to attributes within the title block of a diagram is described.

The structure and syntax of the configuration files is described in Appendix D, Internal Variables and Environment Variables. For sample configuration files see Section 3.4, Configuration Tutorial.

3.1.4 Structure of Format Libraries

You must prepare diagram formats to be used within a project within project-specific format libraries. Formats are handled within a project corresponding the naming conventions:

${C_PROJ_TPL_CAD}/<*applic*>*[*<*standard*>*][*<*language*>*]*/format/..
..<*sublib*>/<*format*>.dwg

Allowed values for <*applic*> are:

- `gef`: for general diagram formats
- `cif`: for circuit diagram formats
- `laf`: for layout diagram formats
- `hof`: for hookup diagram format
- `pif`: for P&I diagram formats
- `gef`: general application independent formats.

Allowed values for <*standard*> are ‘iec’, ‘din’ and ‘cus’.

NOTE

If there are formats independent from any standard, <*standard*> may be omitted.

Allowed values for <*language*> are ‘en’, ‘ge’ and ‘se’.

NOTE

If there are formats independent from any language, <*language*> may be omitted.

Allowed values for <*format*> are ‘a0l’, ‘a1l’, ‘a2l’, ‘a3l’, ‘a3p’, ‘a4l’, ‘a4p’ and ‘q4p’.

The value for <sublib> is not fixed, but is set by creation function to <*applic*><*standard*><*language*> to distinguish between different format libraries allowed in parallel.

Examples for format library structures:

```
${C_PROJ_TPL_CAD}/gefiecen/format/gefiecen/<format>.dwg
${C_PROJ_TPL_CAD}/gefcusse/format/gefcusse/<format>.dwg
```

3.1.5 Handling of Logos in Format Library

Logos are handled in the same way as other symbols, but are included in the sublibrary ‘logos’ within the format library.

${C_PROJ_TPL_DIB}/<*applic*><*standard*><*language*>/format/<*sublib*>/<*format*>.dwg
/blksld/<*sublib*>/<*logoname*>.dwg.

You can build Logo names using the following naming conventions, Figure 3-2:

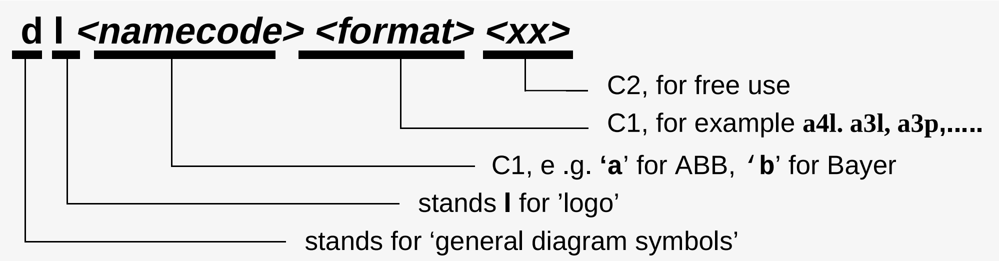

d l <namecode> <format> <xx>
C2, for free use
C1, for example a4l. a3l, a3p,.....
C1, e .g. 'a' for ABB, 'b' for Bayer
stands l for 'logo'
stands for 'general diagram symbols'

Figure 3-2. Naming convention for logos

Examples:

- `dlaa3l01`: name for ABB logo for A3L format
- `dlba4p`: name for Bayer logo for A4P format.

3.1.6 Structure of Project Symbol Libraries

Symbol blocks to be used within a project must be prepared within project-specific symbol libraries.

Symbol blocks are handled within a project corresponding to the naming conventions:

${C_PROJ_LIB_CAD}/*<applic>[<standard>][<language>]*/blksld/*<sublib>*/.....
.....*<symbol>*.dwg
.....*<symbol>*.slb

Allowed values for *<applic>*are

- `cis`: for CIrcuit diagram Symbols
- `las`: for LAyout Symbols,'las'
- `pis`: for P&I diagram Symbols,
- `drs`: for DRawing Symbols
- `hos`: for HOokup diagram Symbols
- `ots`: for OTher/general Symbols.

Allowed values for *<standard>* are `iec`, `din` and `cus`.

NOTE

If symbols are standard independent, *<standard>* may be omitted.

Allowed values for *<language>* are `en`, `ge` and `ge`.

NOTE

If symbols are language independent, *<language>* may be omitted.

The values for *<sublib>* are application-specific and must be limited to 8 characters (see Appendix B, Structuring and Naming Conventions for Symbol Libraries).

Examples for symbol library structures:

${C_PROJ_LIB_CAD}/cisiecen/blksld/motors/.......
${C_PROJ_LIB_CAD}/lasse/blksld/bars/.......
${C_PROJ_LIB_CAD}/cis/blksld/qualif/.......

Symbol block files

The symbol block itself, which is used within a diagram, is represented by a .dwg file. Additionally, for each symbol a .sld file (slide) is required which is used to represent the symbol within the Select block slide menu. Within Diagram Builder a functionality exists which creates .sld files from the .dwg files. Furthermore, a .slb file (slide library) is created.

Library block list

On each <*sublib*> directory, an ascii file blklist.txt is required which includes a list of all included symbol blocks. Within Diagram Builder exists the functionality to update this file automatically, if new symbol blocks are added.

Library data

On each <*sublib*> directory, an ascii file libdata.txt is required which includes the sublibrary-specific configuration parameters. The file must not be empty. The functionality to update these parameters exists within the AdvaBuild Diagram Builder.

Library list

On the .../blksld directory, an ascii file liblist.txt is required, which includes a list of all included sublibraries.

3.1.7 Structure of Plotter Settings Directories

The plotter settings are described in files corresponding to the naming conventions:

`${C_CAPE_ALL_MSC_DIB}/plt/plot/`<*paper_size*><*driver_language*>/<*layout_format*>`.pcp`

Values for <*paper_size*> may be ‘a4’, ‘a3’, ‘a2’, ‘a1’, ‘a0’, ‘lt’, ‘le’

NOTE

‘lt’ means US letter, ‘lg’ means US legal

Values for <*driver_language*> may be ‘ps’, ‘hpgl’

Values for <*layout_format*> may be ‘a4l’, ‘a4p’, ‘a3l’, ‘a3p’, ‘a2l’, ‘a1l’, ‘a0l’, ‘q4p’

The plotter settings included in standard Diagram Builder delivery are:

${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a4l.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a4p.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a3l.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a3p.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a2l.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a1l.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/a0l.pcp
${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps/q4p.pcp

3.1.8 Stand-Alone Environment

AutoCAD may be used as a stand alone product if logged in as unix user *acadfree*.

You can use all available AutoCAD functions. However, drawings which will be created from here are not necessarily usable within (E)DIB. The Electrical Diagram Builder requires a special menu and the drawing must be created by use of the respective initial drawing file.

However, a drawing which was created with the Electrical Diagram Builder can not automatically be manipulated by pure AutoCAD. For this the menu file must be changed and all required fonts must be available in this environment.

Changing drawings which are created by use of acadfree is described in Section 5.1, EXPORT/IMPORT of Existing Drawings.

3.2 Capacity and Performance

This section is not applicable.

3.3 Application Start-up

Go through the following procedure before starting the AdvaBuild Diagram Builder for the first time. You should be logged in as user cape into the UNIX environment and should be logged into the Structure Builder with configuration privileges.

1. Create an object of type DRAWING within the tree structure of the Structure Builder. You should give an adequate name to this object and place it underneath the PROJECT object in order to have direct access to it when doing additional configuration work in later project phases.
2. Start the (Electrical) Diagram Builder and configure AutoCAD as described in Section 3.5.4, (Configuring AutoCAD).

   There are two configuration files, one for work with X-Terminals and one for work at the Console.
3. Install the necessary symbol and format libraries and check configuration files. Choose **Configure Diagram Builder** from STB's **Utility** menu. The AdvaBuild Diagram Builder - Configuration window appears.
   a. Check if the file init.log is displayed within the file list. Open the file and check if all files were copied successfully. The most important files are

      ci_dia.dib, la_dia.dib, pi_dia.dib, drawing.dib, hookup.dib

      dib.sql, dib.dmi

      Those files must be visible within the represented file list as well.

   There are some files with names matching 'mk_std_*.sh'. You may want to install some libraries as described there. If you are going to work with the Electrical Diagram Builder, install the specific libraries for the diagram part and the layout part. The installation files have the following format:

      mk_std_<application><standard><language>.sh

See Section 3.4.2, Installing Symbol and Format Libraries in Project for a more detailed description.

b. Execute the shell script (mk_std_gefiecen.sh for all applications using iec and English or mk_std_gefdinen for all applications using din and English) which installs the appropriate format library. Select the associated icon and choose **Execute** from the **Actions** menu. An Action: Execute dialog box appears. Click on the **OK** button. The Execute window appears and the installation process will be started.

c. Do the same for installing the appropriate symbol libraries. Call the shell script mk_std_cisiecen.sh which installs the symbol library in English according to the IEC standard for use in diagrams within objects of type CI_DIA. Call the shell script mk_std_lasiecen.sh which installs the symbol library in English according to the IEC standard for use in diagrams within objects of type LA_DIA.

How to deal with installation errors is described in Section 3.4.2, Installing Symbol and Format Libraries in Project.

Following these steps, the (Electrical) Diagram Builder should be configured properly. You may want to do a simple test to insure that your installation is correct.

3.4 Configuration Tutorial

This tutorial describes how to configure the (Electrical) Diagram Builder.

3.4.1 Installing DIB/(E)DIB

Either the Diagram Builder (DIB) or the Electrical Diagram Builder (EDIB) must be installed during the setup procedure of AdvaBuild. See the *Advant Station Engineering Station Release Notes R2.1/0* for details.

After installation of AutoCAD and either DIB or EDIB, AutoCAD should be licenced and two AutoCAD configuration files (acad.cfg_<workstation name>) should exist on directories:

- ${C_CAPE_ALL_CFG_DIB}/mouse
  for use from X-Terminals
- ${C_CAPE_ALL_CFG_DIB}/digit
  for use from the Console.

3.4.2 Installing Symbol and Format Libraries in Project

3.4.2.1 Setup for configuration

1. Login to AS500ES as user **`cape.`** This is necessary because only this user has privileges to create libraries.
2. Start STB, and login as user with configure privileges.
3. Choose **Configure Diagram Builder** from STB's **Utility** menu. The configuration utility for the Diagram Builder comes up and shows all project specific Diagram Builder related configuration files, Figure 3-3.

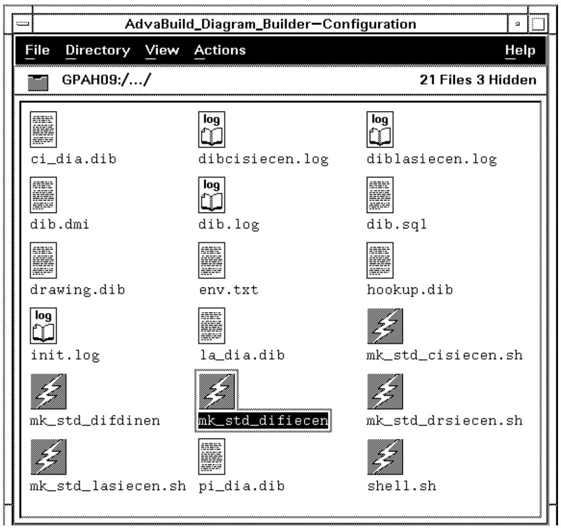


Figure 3-3. AdvaBuild Diagram Builder Configuration Menu

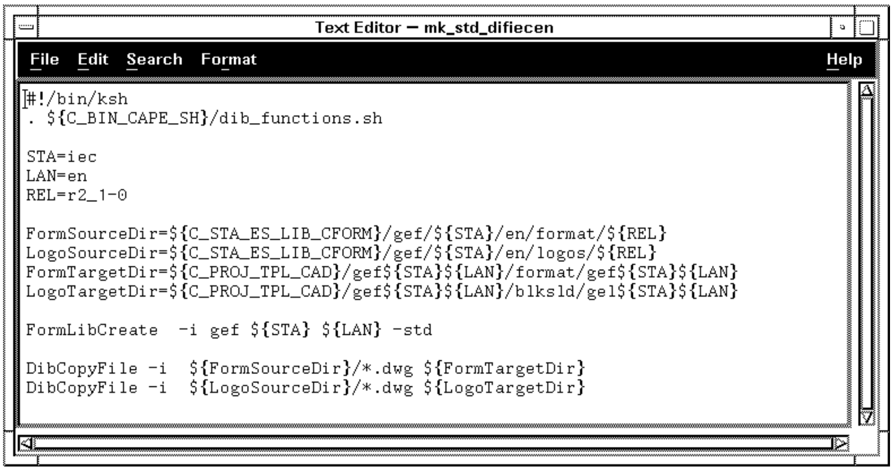
3.4.2.2 Installing symbol and format libraries in project

4. Create shell script file. When this utility is opened the first time, some example scripts (starting with `mk_std_...`) for library installation are copied from the delivered symbol and format libraries (see Section 3.3, Application Start-up). These can be used directly for installing format and symbol libraries into the project or can be used as examples and adapted to project specific needs.

   Format libraries can be used by all Diagram Builder applications and need therefore be installed normally only once within a project. Symbol libraries are normally specific for each application and are therefore installed application specific.

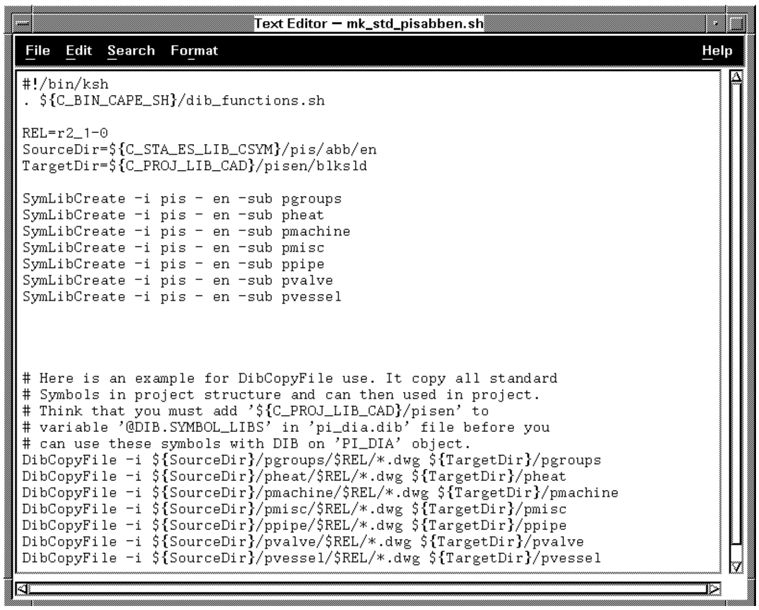
   It is also possible to install within a project in parallel several format and symbol libraries according to different standards (IEC,DIN, CUStomer) and languages (EN, GE, SE), if there are libraries available. Configuration and setup control for the different Diagram Builder applications is explained in step 7.

5. Edit shell script file. For editing shell scripts, select and choose **Actions Edit** from DIB **Configuration Utility** menu.



```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/dib_functions.sh

STA=iec
LAN=en
REL=r2_1-0

FormSourceDir=${C_STA_ES_LIB_CFORM}/gef/${STA}/en/format/${REL}
LogoSourceDir=${C_STA_ES_LIB_CFORM}/gef/${STA}/en/logos/${REL}
FormTargetDir=${C_PROJ_TPL_CAD}/gef${STA}${LAN}/format/gef${STA}${LAN}
LogoTargetDir=${C_PROJ_TPL_CAD}/gef${STA}${LAN}/blksld/gel${STA}${LAN}

FormLibCreate  -i gef ${STA} ${LAN} -std

DibCopyFile -i  ${FormSourceDir}/*.dwg ${FormTargetDir}
DibCopyFile -i  ${LogoSourceDir}/*.dwg ${LogoTargetDir}
```

Figure 3-4. Example for IEC format and logo library installation script



```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/dib_functions.sh

REL=r2_1-0
SourceDir=${C_STA_ES_LIB_CSYM}/pis/abb/en
TargetDir=${C_PROJ_LIB_CAD}/pisen/blksld

SymLibCreate -i pis - en -sub pgroups
SymLibCreate -i pis - en -sub pheat
SymLibCreate -i pis - en -sub pmachine
SymLibCreate -i pis - en -sub pmisc
SymLibCreate -i pis - en -sub ppipe
SymLibCreate -i pis - en -sub pvalve
SymLibCreate -i pis - en -sub pvessel




# Here is an example for DibCopyFile use. It copy all standard
# Symbols in project structure and can then used in project.
# Think that you must add '${C_PROJ_LIB_CAD}/pisen' to
# variable '@DIB.SYMBOL_LIBS' in 'pi_dia.dib' file before you
# can use these symbols with DIB on 'PI_DIA' object.
DibCopyFile -i ${SourceDir}/pgroups/$REL/*.dwg ${TargetDir}/pgroups
DibCopyFile -i ${SourceDir}/pheat/$REL/*.dwg ${TargetDir}/pheat
DibCopyFile -i ${SourceDir}/pmachine/$REL/*.dwg ${TargetDir}/pmachine
DibCopyFile -i ${SourceDir}/pmisc/$REL/*.dwg ${TargetDir}/pmisc
DibCopyFile -i ${SourceDir}/ppipe/$REL/*.dwg ${TargetDir}/ppipe
DibCopyFile -i ${SourceDir}/pvalve/$REL/*.dwg ${TargetDir}/pvalve
DibCopyFile -i ${SourceDir}/pvessel/$REL/*.dwg ${TargetDir}/pvessel
```

Figure 3-5. Example for symbol library installation script

6. Execute shell script file. Scriptfiles are executed by mouse double-click on file icon.

CAUTION

Be sure Filemanager was not started before in other environment within session.
If so, close down session completely or kill vuefile process on Unix level.

7. Adapt library setup and printer definitions in configuration file. For each Diagram Builder application, a separate project specific configuration file exists on the DIB configuration directory. Those files are:
- ci_dia.dib : for CI_DIA objects setup
- la_dia.dib: for LA_DIA objects setup

- pi_dia.dib : for PI_DIA objects setuI
- hookup.dib: for HOOKUP objects setup
- drawing.dib : for DRAWING objects setup

For library setup the following setup parameters are relevant:

@ACAD: will be mapped to ACAD path, must include, for example, full directory path definitions for all libraries including symbols, which can be inserted directly by the AutoCAD insert function.

NOTE

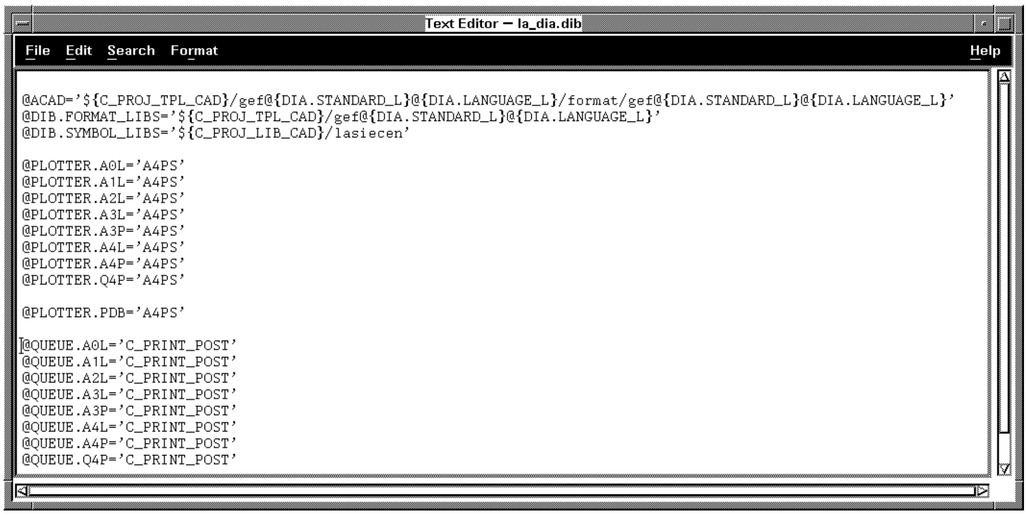
For revision block handling, the directory including the revision blocks must also be referenced here!

@DIB.FORMAT_LIBS: must include the directory path to all format libraries, which can be used within (E)DIB via **Format Insert** menu item. It is possible to reference several format libraries.

@DIB.SYMBOL_LIBS: must include the directory path to all symbol libraries, which can be used within (E)DIB via Symbol within the **Symlib Insert** menu item. It is possible to reference up to two symbol libraries. For (E)DIB it is important to reference the default library as first!

If libraries with different standards or languages are used within one project, the correct standard and language specific setup values from the diagram representing the data base object can be used here (see setup for data base select via dib.sql file). An adaptive setup can be configured as a result.

Additionally, the frame format specific selection of plotter configuration files and plotter queues can be configured here.



```
@ACAD='${C_PROJ_TPL_CAD}/gef@{DIA.STANDARD_L}@{DIA.LANGUAGE_L}/format/gef@{DIA.STANDARD_L}@{DIA.LANGUAGE_L}'
@DIB.FORMAT_LIBS='${C_PROJ_TPL_CAD}/gef@{DIA.STANDARD_L}@{DIA.LANGUAGE_L}'
@DIB.SYMBOL_LIBS='${C_PROJ_LIB_CAD}/lasiecen'

@PLOTTER.A0L='A4PS'
@PLOTTER.A1L='A4PS'
@PLOTTER.A2L='A4PS'
@PLOTTER.A3L='A4PS'
@PLOTTER.A3P='A4PS'
@PLOTTER.A4L='A4PS'
@PLOTTER.A4P='A4PS'
@PLOTTER.Q4P='A4PS'

@PLOTTER.PDB='A4PS'

@QUEUE.A0L='C_PRINT_POST'
@QUEUE.A1L='C_PRINT_POST'
@QUEUE.A2L='C_PRINT_POST'
@QUEUE.A3L='C_PRINT_POST'
@QUEUE.A3P='C_PRINT_POST'
@QUEUE.A4L='C_PRINT_POST'
@QUEUE.A4P='C_PRINT_POST'
@QUEUE.Q4P='C_PRINT_POST'
```

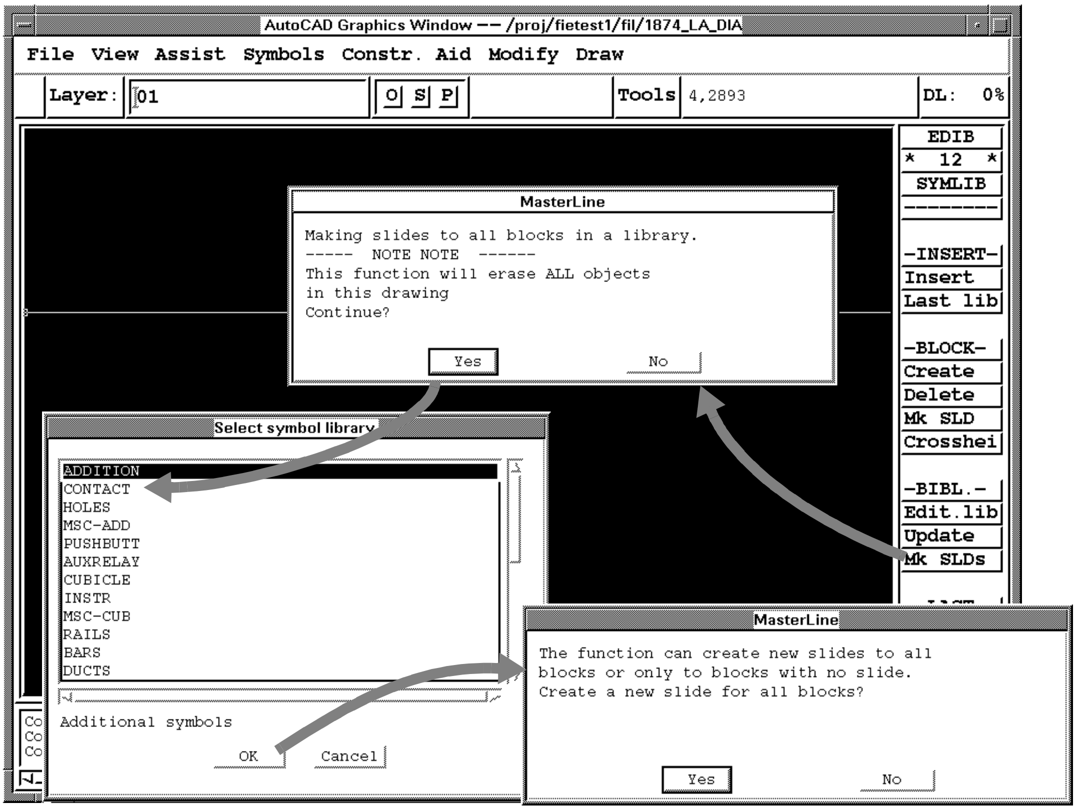
Figure 3-6. Example Plotter Configuration and Queues Script

3.4.2.3 Generating slide menus for symbol sublibrary

8. Insert a dummy diagram object of corresponding type via STB. If STANDARD and LANGUAGE is used for symbol library setup, enter corresponding values via TEB.

9. Open the Diagram view on the selected diagram object. Do not insert any format when (E)DIB comes up.

10. Choose **SYMLIB Mk SLDs** from right hand side (E)DIB menu. Agree to erasing all objects in drawing. Select sublibrary and start creating slides.



Making slides to all blocks in a library.
----- NOTE NOTE ------
This function will erase ALL objects
in this drawing
Continue?

The function can create new slides to all
blocks or only to blocks with no slide.
Create a new slide for all blocks?

Figure 3-7. Side Menu Generation

11. Repeat step before for all sublibraries for which slides are required. Here, slides can be generated for logos. But this needs only to be done once.

12. Repeat steps 8 to 11 for all applications.

NOTE

Do not use the current drawing as it is for a later purpose, as all information was deleted, including important configuration information. You may leave (E)DIB without saving the changes or remove the drawing from within the Structure Builder.

3.4.3 Adaptation of title block mapping in Project

3.4.3.1 Setup for configuration

1. Start STB, login as user with configure privileges
2. Choose **Configure Diagram Builder** from STB's **Utility** menu.
   The configuration utility for the Diagram Builder comes up and shows all project specific Diagram Builder related configuration files.

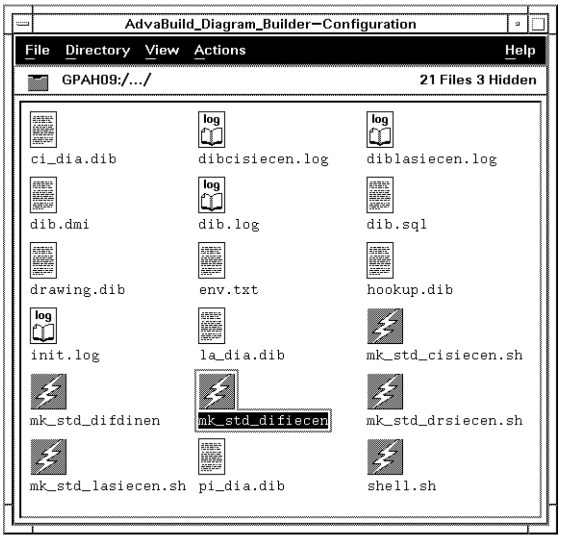


Figure 3-8. Configuration Setup

NOTE

For adapting the configuration files, you must be familiar with SQL syntax and handling of environment variables within HP-UX.

3.4.3.2 Configuration of file dib.sql

1. Click on the icon for file dib.sql.
2. Choose menu item **Edit** from the **Actions** menu.
3. The Text Editor appears and displays the file dib.sql. A detailed description of the file follows.

3.4.3.3 Configuration of file dib.dmi

1. Click on the icon for file dib.dmi.
2. Choose menu item **Edit** from the **Actions** menu.
3. The Text Editor appears and displays the file dib.dmi. A detailed description of the file follows.

```
SELECT
```

Date conversion

```
’DIA.LM_DATE_A=’  || TO_CHAR(OI.MOD_DATE,’YY-MM-DD’)          || ’<!>’ ,
’DIA.LM_DATE_B=’  || TO_CHAR(OI.MOD_DATE,’DD.MM.YY’)          || ’<!>’ ,
’DIA.LM_DATE_C=’  || TO_CHAR(OI.MOD_DATE,’YYYYMMDD’)          || ’<!>’ ,
’SYS.PR_DATE_A=’  || TO_CHAR(SYSDATE,’YY-MM-DD’)              || ’<!>’ ,
’SYS.PR_DATE_B=’  || TO_CHAR(SYSDATE,’DD.MM.YY’)              || ’<!>’ ,
’SYS.PR_DATE_C=’  || TO_CHAR(SYSDATE,’YYYYMMDD’)              || ’<!>’ ,
```

```
’DIA.FILE=’       || SUBSTR(OID.FILE_NAME,1,LENGTH(OID.FILE_NAME)-4) || ’<!>’ ,
’DIA.OBJ_ID=’     || NVL(OI.OBJ_ID,’’)                        || ’<!>’ ,
’DIA.SYM_NAME=’   || NVL(OI.SYM_NAME,’’)                      || ’<!>’ ,
’DIA.C_P_TREE_ID=’|| NVL(OI.C_P_TREE_ID,’’)                   || ’<!>’ ,
’DIA.ITEM_DES_2=’ || NVL(OID.ITEM_DES_2,’’)                   || ’<!>’ ,
’DIA.NO_OF_SH=’   || NVL(OID.NO_OF_SH,’’)                     || ’<!>’ ,
’DIA.DOC_TYPE=’   || NVL(OID.DOC_TYPE,’’)                     || ’<!>’ ,
’DIA.DOC_TYPE_1=’ || @{DIB.DOC_TYPE_1}                        || ’<!>’ ,
’DIA.DOC_TYPE_2=’ || @{DIB.DOC_TYPE_2}                        || ’<!>’ ,
’DIA.DOC_DES=’    || NVL(OID.DOC_DES,’’)                      || ’<!>’ ,
```

Default values

```
’DIA.LANGUAGE=’   || NVL(OID.LANGUAGE,’EN’)                   || ’<!>’ ,
’DIA.LANGUAGE_L=’ || NVL(LOWER(OID.LANGUAGE),’en’)            || ’<!>’ ,
’DIA.STANDARD=’   || NVL(OID.STANDARD,’IEC’)                  || ’<!>’ ,
’DIA.STANDARD_L=’ || NVL(LOWER(OID.STANDARD),’iec’)           || ’<!>’ ,
```

```
’DIA.PREP_DATE=’  || NVL(OID.PREP_DATE,’’)                    || ’<!>’ ,
’DIA.PREP_DATE_IEC=’ || TO_CHAR(NVL(OID.PREP_DATE,’’),’YY-MM-DD’) || ’<!>’ ,
’DIA.PREP_DATE_DIN=’ || TO_CHAR(NVL(OID.PREP_DATE,’’),’DD.MM.YY’) || ’<!>’ ,
’DIA.PREP_NAME=’  || NVL(OID.PREP_NAME,’’)                    || ’<!>’ ,
’DIA.CHK_DATE=’   || NVL(OID.CHK_DATE,’’)                     || ’<!>’ ,
```

Variant handling

```
’DIA.CHK_DATE_IEC=’ || TO_CHAR(NVL(OID.CHK_DATE,’’),’YY-MM-DD’) || ’<!>’ ,
’DIA.CHK_DATE_DIN=’ || TO_CHAR(NVL(OID.CHK_DATE,’’),’DD.MM.YY’) || ’<!>’ ,
```

```
’DIA.CHK_NAME=’   || NVL(OID.CHK_NAME,’’)                     || ’<!>’ ,
’DIA.APPR_DATE=’  || NVL(OID.APPR_DATE,’’)                    || ’<!>’ ,
’DIA.APPR_DATE_IEC=’ || TO_CHAR(NVL(OID.APPR_DATE,’’),’YY-MM-DD’) || ’<!>’ ,
’DIA.APPR_DATE_DIN=’ || TO_CHAR(NVL(OID.APPR_DATE,’’),’DD.MM.YY’) || ’<!>’ ,
’DIA.APPR_NAME=’  || NVL(OID.APPR_NAME,’’)                    || ’<!>’ ,
’DIA.BASED_ON=’   || NVL(OID.BASED_ON,’’)                     || ’<!>’ ,
’DIA.REPLACES=’   || NVL(OID.REPLACES,’’)                     || ’<!>’ ,
’DIA.REPL_BY=’    || NVL(OID.REPL_BY,’’)                      || ’<!>’ ,
’DIA.FREE_DIS=’   || NVL(OID.FREE_DIS,’’)
```

Using Variables

```
FROM
   OBJ_INSTANCE OI,
   @{DIB.OBJ_TYPE:-} OID
WHERE
   OID.OBJ_ID=’@{DIB.OBJ_ID:-}’AND
   OI.C_INT_KEY = OID.C_INT_KEY
END
```

Figure 3-9. Selection of Diagram Object Data

- **Date conversion.** Use the function `TO_CHAR(<data field>, <format>)` to change a selected date’s format. To select several variants at a time, assign them to different variable names.
- **Default values.** You can handle null values within the data base with the function `NVL(<data filed>,<default value>)`. If no value will be selected, the value specified as default will be returned instead.
- **Variant handling.** To make variant handling possible, two or more versions of the same item must be selected. Add the text that is used to choose the variant as part of the variable name. See Figure 3-13 for a description of the dependencies. The goal is to use values of other variables as the selectors of the valid variant. For example,

@{DIA.CHK_DATE_@{DIA.STANDARD}}).

```
                     SELECT
                     ’P.OBJ_ID=’       ||  NVL(P_OI.OBJ_ID,’’)       || ’<!>’ ,
Parent Object        ’P.SYM_NAME=’     ||  NVL(P_OI.SYM_NAME,’’)     || ’<!>’ ,
                     ’P.OBJ_TYPE=’     ||  NVL(P_OI.OBJ_TYPE,’’)     || ’<!>’ ,
                     ’PP.OBJ_ID=’      ||  NVL(PP_OI.OBJ_ID,’’)      || ’<!>’ ,
Grand Parent         ’PP.SYM_NAME=’    ||  NVL(PP_OI.SYM_NAME,’’)    || ’<!>’ ,
Object               ’PP.OBJ_TYPE=’    ||  NVL(PP_OI.OBJ_TYPE,’’)    || ’<!>’ ,
                     ’PPP.OBJ_ID=’     ||  NVL(PPP_OI.OBJ_ID,’’)     || ’<!>’ ,
                     ’PPP.SYM_NAME=’   ||  NVL(PPP_OI.SYM_NAME,’’)   || ’<!>’ ,
Great Grand          ’PPP.OBJ_TYPE=’   ||  NVL(PPP_OI.OBJ_TYPE,’’)
Parent Object        FROM
                        OBJ_INSTANCE P_OI,
                        OBJ_INSTANCE PP_OI,
                        OBJ_INSTANCE PPP_OI
                     WHERE
                        P_OI.C_TREE_ID = ’@{DIA.C_P_TREE_ID:-}’AND
                        P_OI.C_P_TREE_ID = PP_OI.C_TREE_ID (+) AND
Outer Join              PP_OI.C_P_TREE_ID = PPP_OI.C_TREE_ID (+)
                     END
```

Figure 3-10. Variant Handling

- **Using Variables.** Use internal variables at any place of this file. You can use predefined variables such as `DIB.OBJ_TYPE` and `DIB.OBJ_ID`. Used this way, variables make your select statement more flexible. To make your select statement more readable, use aliases for table names.
- **Parent Object.** Some attributes of the Parent Object will be selected.
- **Grand Parent Object.** Some attributes of the Grand Parent Object will be selected if it exists. If the current object is placed directly underneath the Project Object, it does not have a grand parent.
- **Great Grand Parent Object.** Some attributes of the Great Grand Parent Object will be selected if it exists. If the current object is placed directly underneath the Project Object or one level deeper, it does not have a great grand parent.
- **Outer Join.** The Outer Join combines different tables, or many versions of the same table, even if there are no rows that fulfill the join criteria. If one self-joins the table OBJ_INSTANCE, as shown above, three times via attributes C_TREE_ID (the tree id of the current object/row) and C_P_TREE_ID (the tree id of the parent object) there may not be enough parents to get all of the required attributes. In this case, NULL values will be returned by the select statement and no unexpected error occurs.

```
                   SELECT
                   'PROJECT.PRO_NAME_1='      || NVL(PRO_NAME_1,'')      || '<!>' ,
                   'PROJECT.PRO_NAME_2='      || NVL(PRO_NAME_2,'')      || '<!>' ,
                   'PROJECT.PRO_NAME_3='      || NVL(PRO_NAME_3,'')      || '<!>' ,
                   'PROJECT.RESP_DEPT='       || NVL(RESP_DEPT,'')       || '<!>' ,
                   'PROJECT.CUSTOMER='        || NVL(CUSTOMER,'')        || '<!>' ,
Priority           'PROJECT.FACTORY='         || NVL(FACTORY,'')         || '<!>' ,
Handling           'PROJECT.OWNER='           || NVL(OWNER,'')           || '<!>' ,
                   'PROJECT.PRO_NO='          || NVL(PRO_NO,'')
                   FROM     PROJECT
                   WHERE
                            C_P_TREE_ID = '0'
                   END

                   @{.CalledBy(DOB):+SELECT:-NO SELECT}
Conditional        'DSTRUC.OBJ_ID='           || NVL(OBJ_ID,'')          || '<!>' ,
Execution          'DSTRUC.OWNER='            || NVL(OWNER,'')           || '<!>' ,
Priority           'DSTRUC.CUST_DIS='         || NVL(CUST_DIS,'')
Handling           FROM     DSTRUC
                   WHERE
                            OBJ_ID = '@{PDB.DSTRUC.OBJ_ID:-}'
                   END
```

Figure 3-11. Priority Handling and Conditional Execution

- **Conditional Execution.** Before the SQL statement is executed all variables will be expanded. The construction `@{.CalledBy(DOB):+SELECT:-NO SELECT`) introduces a select statement only if the variable/predicate `.CalledBy(DOB)` was a defined value, thus if the (E)DIB was called by the Plant Documentation Builder. The variable/predicate is predefined. It is also possible to use `.CalledBy(STB)` if you want to recognize that the Diagram Builder is called by the Structure Builder and not by the Plant Documentation Builder (DOB - for historical reasons).
- **Priority Handling.** Sometimes you may want to select two different versions of attributes coming from different contexts (for example, PROJECT object or DSTRUC object). There is a way to define some kind of priority handling which describes which one should be taken rather than the other one. See below for details.

```
                  DIB_VERSION                        DOC_NAME_2
                  @{DIB.PROG_VERSION_TEXT}           @{DIA.DOC_TYPE_2:-}
                  DOC_NO                             LANGUAGE
                  @{DSTRUC.OBJ_ID:-}                 @{DIA.LANGUAGE:-}
                  OWNER                              PREP_DATE
Priority          @{@{IsEmptyStr@{DSTRUC.OWNER:      @{DIA.PREP_DATE_@{DIA.STANDARD:-}:-}
Handling          -}:+PROJECT.OWNER:-                PREP_NAME
                  DSTRUC.OWNER}:-}                   @{DIA.PREP_NAME:-}
Variant           CUST_DIS                           CHK_DATE
Handling          @{DSTRUC.CUST_DIS:-}               @{DIA.CHK_DATE_@{DIA.STANDARD:-}:-}
                  PRO_NAME_1                         CHK_NAME
                  @{PROJECT.PRO_NAME_1:-}            @{DIA.CHK_NAME:-}
                  PRO_NAME_2                         APPR_DATE
                  @{PROJECT.PRO_NAME_2:-}            @{DIA.APPR_DATE_@{DIA.STANDARD:-}:-}
                  PRO_NAME_3                         APPR_NAME
                  @{PROJECT.PRO_NAME_3:-}            @{DIA.APPR_NAME:-}
                  RESP_DEPT                          BASED_ON
                  @{PROJECT.RESP_DEPT:-}             @{DIA.BASED_ON:-}
                  CUSTOMER                           REPLACES
                  @{PROJECT.CUSTOMER:-}              @{DIA.REPLACES:-}
                  FACTORY                            REPL_BY
                  @{PROJECT.FACTORY:-}               @{DIA.REPL_BY:-}
                  PRO_NO                             FREE_DIS
                  @{PROJECT.PRO_NO:-}                @{DIA.FREE_DIS:-}
Default           NO_OF_SH                           STANDARD
Value             @{DIA.NO_OF_SH:-}                  @{DIA.STANDARD:-}
                  SHEET                              FILE
                  @{DIB.SHNOBLK:-}                   @{DIA.FILE:-}
                  ITEM_DES_1                         TITLE_1
                  @{DIB.ITEMDES:-}                   @{DIA.SYM_NAME:-}
                  ITEM_DES_2                         TITLE_2
                  @{DIA.ITEM_DES_2:-}                @{P.SYM_NAME:-}
                  DOC_KIND                           TITLE_3
                  @{DIB.DKCODE:-}                    @{PP.SYM_NAME:-}
                  DOC_NAME_1                         TITLE_4
                  @{DIA.DOC_TYPE_1:-}                @{PPP.SYM_NAME:-}
```

Figure 3-12. Title Block Attribute Mapping

- **Priority Handling.** The expression

  `@{@{IsEmptyStr@{DSTRUC.OWNER:-}:+PROJECT.OWNER:-DSTRUC.OWNER}:-}`

  formulates some kind of priority handling. You can write this expression as

  ```
  if DSTRUC.OWNER is an empty string
     then      expand PROJECT.OWNER
     else      expand DSTRUC.OWNER
  ```

  This means that the OWNER attribute will be taken from the DSTRUC object if it is not empty. Only in the other case the OWNER attribute is taken from the PROJECT object.

  You can use the construct

  ```
  @{                             @{IsEmptyStr@{<variable name>:-
  }:+<true case>:-<false case>}:-}
  ```

  to define different behavior for the case that a string is empty or not.

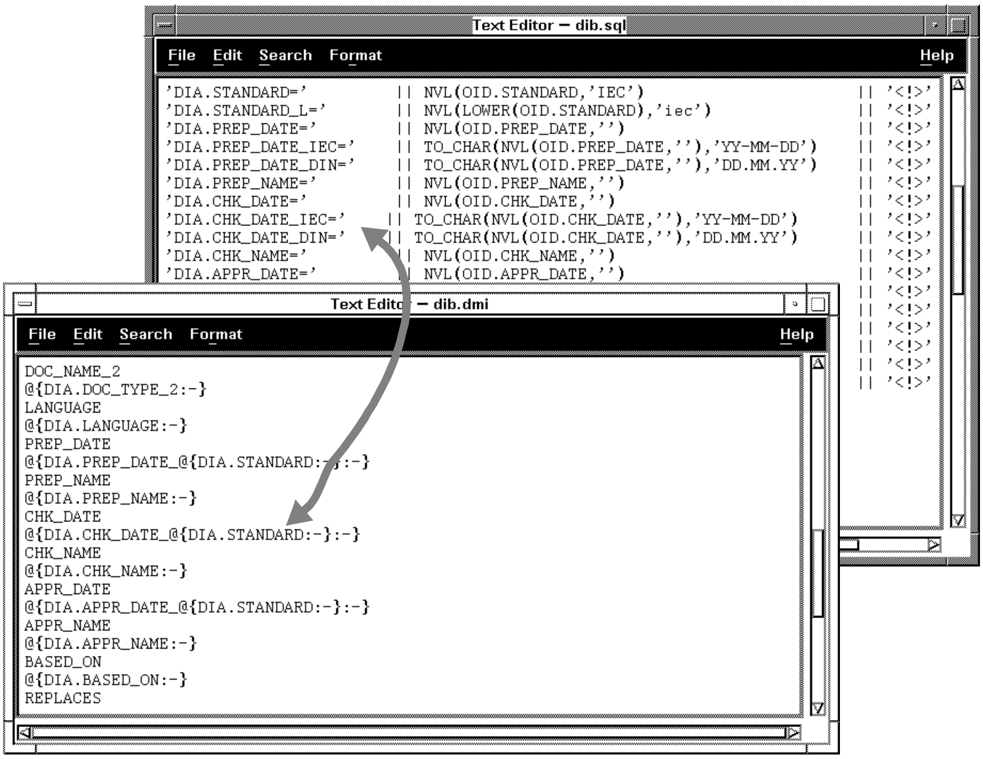
- **Variant Handling.** You can use different date formats to make one of them dependent on some other attribute value. To do this, use variables recursively:

  `@{DIA.CHK_DATE_@{DIA.STANDARD:-}:-}`

  The example uses the value which will be expanded by `@{DIA.STANDARD:-}`to verify which variant should be taken:

  `DIA.CHK_DATE_IEC` or `DIA.CHK_DATE_DIN.`

  See also Figure 3-13 which illustrates the dependencies sql-file and dmi-file.



```
'DIA.STANDARD='         || NVL(OID.STANDARD,'IEC')                         || '<!>'
'DIA.STANDARD_L='       || NVL(LOWER(OID.STANDARD),'iec')                  || '<!>'
'DIA.PREP_DATE='        || NVL(OID.PREP_DATE,'')                           || '<!>'
'DIA.PREP_DATE_IEC='    || TO_CHAR(NVL(OID.PREP_DATE,''),'YY-MM-DD')       || '<!>'
'DIA.PREP_DATE_DIN='    || TO_CHAR(NVL(OID.PREP_DATE,''),'DD.MM.YY')       || '<!>'
'DIA.PREP_NAME='        || NVL(OID.PREP_NAME,'')                           || '<!>'
'DIA.CHK_DATE='         || NVL(OID.CHK_DATE,'')                            || '<!>'
'DIA.CHK_DATE_IEC='    || TO_CHAR(NVL(OID.CHK_DATE,''),'YY-MM-DD')         || '<!>'
'DIA.CHK_DATE_DIN='    || TO_CHAR(NVL(OID.CHK_DATE,''),'DD.MM.YY')         || '<!>'
'DIA.CHK_NAME='         || NVL(OID.CHK_NAME,'')                            || '<!>'
'DIA.APPR_DATE='        || NVL(OID.APPR_DATE,'')                           || '<!>'
```

```
DOC_NAME_2
@{DIA.DOC_TYPE_2:-}
LANGUAGE
@{DIA.LANGUAGE:-}
PREP_DATE
@{DIA.PREP_DATE_@{DIA.STANDARD:-}:-}
PREP_NAME
@{DIA.PREP_NAME:-}
CHK_DATE
@{DIA.CHK_DATE_@{DIA.STANDARD:-}:-}
CHK_NAME
@{DIA.CHK_NAME:-}
APPR_DATE
@{DIA.APPR_DATE_@{DIA.STANDARD:-}:-}
APPR_NAME
@{DIA.APPR_NAME:-}
BASED_ON
@{DIA.BASED_ON:-}
REPLACES
```

Figure 3-13. Dependencies between dib.sql and dib.dmi

- **Default Value.** Default values are chosen within this file to create a fault tolerant situation. If a certain variable is not defined, this will lead to an error and yield an interruption of an open or print process. You may want to know about nondefined values within your title block. In this case you may drop the default value by dropping the term ‘`:-`’ from, for example, `@{DIB.ITEMDES:-}`or you may place a readable default value as in `@{DIB.ITEMDES:-<<undefined>>}.` Now you will be informed within your printouts about missing values. This might be of interest because the value will only be defined if the object identification (OBJ_ID) of the corresponding diagram object is set correctly.
  (See Section 4.4.3.1, Inserting Diagram Objects in AdvaBuild for explanations.)

- **Guarded Values.** To prevent some values within the title block from updating by values coming from the Diagram Builder, drop both lines within the .dmi file, the attribute name line and the attribute value line. To specify the first line of the drawing title within the drawing, drop the following two lines from the .dmi file.

TITLE_1
@{DIA.SYM_NAME:-}

You can use variables within variable names. This makes it possible to define variants for title block mapping of some degree. Within the delivered standard configuration, this is used to handle different date formats depending from standard.

3.5 Configuration Procedures

You can configure the (Electrical) Diagram Builder by choosing STB's menu items **Utility Configure Diagram Builder**.

This will start the VUE File Manager on the project specific Diagram Builder configuration directory.

NOTE

This can only be done by a user with configure privileges logged in to AS500ES with user name cape.

3.5.1 Symbol and Format Library handling functions

Symbol and format directory handling is supported by specific shell functions on the operating system level.

You can execute shell functions within shellscript files, which can be created, edited and executed by VUE File Manager on the Diagram Builder configuration directory. Include the following two lines as the first two lines of your shell script. The first line guarantees that the shell will be called and the second line includes a set of functions to be used for installation purposes:

```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/c_dib_functions.sh
```

3.5.1.1 Creating Symbol Libraries

Use the following shell function to create symbol libraries within a project:

SymLibCreate [-i|-f] <*application*> <*standard*> <*language*> [-std | -sub <*sublibname*>]

The optional controlling attributes mean:

- no option set: existing directories are not overwritten, not existing are created
- -i: stands for interactive/inquire, asks if existing directories should be overwritten
- -f: stands for force, existing directories are overwritten (cleared) without asking.

Allowed values fur <*application*> are:

- cis: for CIrcuit diagram Symbols
- las: for LAyout Symbols,'las'
- pis: for P&I diagram Symbols,
- drs: for DRawing Symbols
- hos: for HOokup diagram Symbols
- ges: for general Symbols.

Allowed values for <*standard*> are:

- iec: for IEC standard
- din: for DIN standard
- cus: for customer standard.

NOTE

Use '-' when symbols are independent from standard.

Allowed values for <*language*> are:

- en: for english
- ge: for german
- se: for swedish.

NOTE

Use '-' when symbols are independent from language.

The following different alternatives are provided for creating the required library structures:

1. SymLibCreate <*application*> <*standard*> <*language*>
Creates an empty symbol library without sublibrary directories within project identified by <application><standard><language>
Examples:

SymLibCreate cis iec en	creates *${C_PROJ_LIB_DIB}/cisiecen/..*
SymLibCreate cis iec -	creates *${C_PROJ_LIB_DIB}/cisiec/..*
SymLibCreate cis - en	creates *${C_PROJ_LIB_DIB}/cisen/..*
SymLibCreate cis - -	creates *${C_PROJ_LIB_DIB}/cis/..*

2. SymLibCreate <*application*> <*standard*> <*language*> -std
Creates an empty symbol library with all standard sublibrary directories corresponding <application> within project identified by <application><standard><language>
Examples:

SymLibCreate cis iec en -std	creates *${C_PROJ_LIB_DIB}/cisiecen/...* plus sublibraries for circuit diagram symbols
SymLibCreate las - en -std	creates *${C_PROJ_LIB_DIB}/lasen/..* plus sublibraries for layout diagram symbols

3. SymLibCreate <*application*> <*standard*> <*language*> -sub <sublibname>
Creates an empty sublibrary identified by <sublibname> underneath the symbol library identified by <application><standard><language>.
If the corresponding symbol library does not exist, it will also be created.
Examples:

SymLibCreate drs - - -sub privat	creates *${C_PROJ_LIB_DIB}/drs/blksld/privat*

Example shellscript file for symbol library creation:
Filename: make_symlibs.sh

```
#!/bin/ksh
 ${C_BIN_CAPE_SH}/c_dib_functions.sh

SymLibCreate cis iec en -std
SymLibCreate las - en -std
SymLibCreate pis iec - -std
SymLibCreate hos - en -std
SymLibCreate drs - en -std
SymLibCreate drs - en -sub privat
```

3.5.1.2 Updating Symbol Libraries

Use the following shell function to update symbol libraries from external symbol libraries:

DibCopyFile [-f|-i] <*source pathname*>/<*filter*> <*destination pathname*>

<*source pathname*> and <*destination pathname*> mean full pathname on UNIX file system.

<filter> corresponds to normal UNIX file name filter mechanisms (e.g use '*' and '?' as wildcards). The full syntax corresponds to UNIX expressions described in "*Rule Qualification for Patterns Used for Filename Expansion*" on manual page (5). (Accessible by typing *man 5 regexp* in hpterm.)

If option -f is set, already existing files will be overwritten. If option -i is set, the user will be asked. If no option is set, already existing files will not be overwritten.

Example 1: Copying only motor and transducer symbols

```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/c_dib_functions.sh

REL=r2_0-1
SLIB=${C_STA_ES_LIB_CSYM}/cis/iec/en
TLIB=${C_PROJ_LIB_CAD}/cisiecen/blksld

DibCopyFile -f ${SLIB}/motors/${REL}/*.dwg ${TLIB}/motors
DibCopyFile -f ${SLIB}/transduc/${REL}/*.dwg ${TLIB}/transduc
```

Example 2: Copying only motor and transducer symbols with 'a' as last character in filename

```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/c_dib_functions.sh

REL=r2_0-1
SLIB=${C_STA_ES_LIB_CSYM}/cis/iec/en
TLIB=${C_PROJ_LIB_CAD}/cisiecen/blksld

DibCopyFile -f ${SLIB}/motors/${REL}/???????a.dwg ${TLIB}/motors
DibCopyFile -f ${SLIB}/transduc/${REL}/???????a.dwg ${TLIB}/transduc
```

It would be safer to copy from the source library at first into a temporary library and copy from there the complete content into the final destination library!

Example 3:

```
REL=r2_0-1
SLIB=${C_STA_ES_LIB_CSYM}/cis/iec/en
TLIB=tmp/dib/cis

mkdir -p ${TLIB}/motors
mkdir -p ${TLIB}/transduc
cp ${SLIB}/motors/${REL}/???????a.dwg ${TLIB}/motors
cp ${SLIB}/transduc/${REL}/???????a.dwg ${TLIB}/trans-duc
```

Inspect the result first, then:

```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/c_dib_functions.sh

SLIB=tmp/dib/cis
TLIB=${C_PROJ_LIB_CAD}/cisiecen/blksld

DibCopyFile -f ${SLIB}/motors/* ${TLIB}/motors
DibCopyFile -f ${SLIB}/transduc/* ${TLIB}/transduc

cd tmp/dib
rm -R cis
```

NOTE

After updating symbol libraries on file system, the slide menu must be updated within the (Electrical) Diagram Builder.

3.5.1.3 Creating Format Libraries

Use the following shell function to create format libraries within a project:

FormLibCreate [-i|-f] <*application*> <*standard*> <*language*>

The optional controlling attributes mean:

- no option set: existing directories are not overwritten, not existing are created
- -i: stands for interactive/inquire, asks if existing directories should be overwritten
- -f: stands for force, existing directories are overwritten (cleared) without asking.

Allowed values for <*application*> are:

- gef: for general Diagram Builder formats
- cif: for circuit diagram formats
- laf: for layout diagram formats
- pif: for P&I diagram formats
- drf: for drawing formats
- hof: for hookup diagram formats.

Allowed values for <*standard*> are:

- iec: for IEC standard
- din: for DIN standard
- cus: for customer standard.

NOTE

Use '**-**' when formats are independent of the standard.

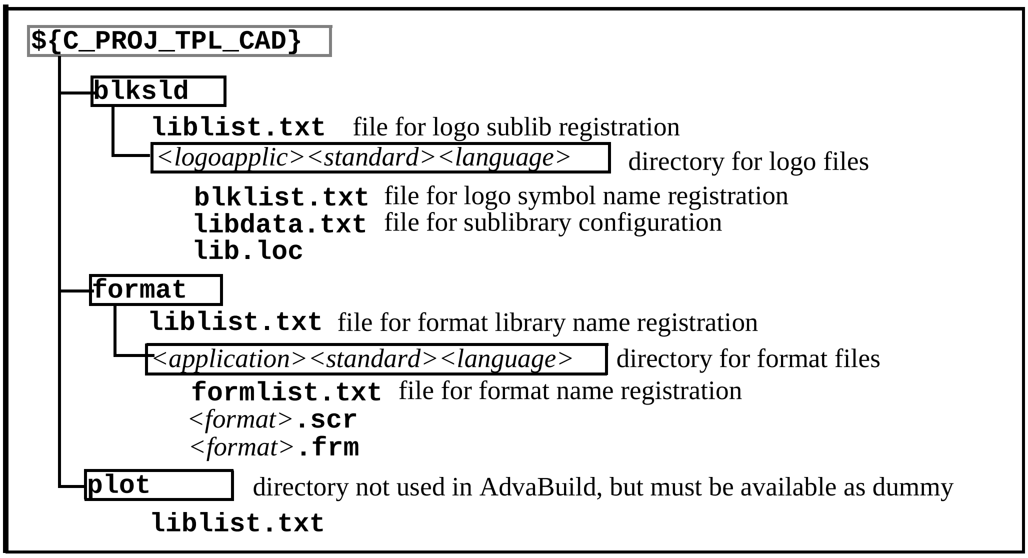
Allowed values for <*language*> are:

- en: for english
- ge: for german
- se: for swedish.

NOTE

Use '**-**' when formats are independent of language.

The FormLibCreate function creates the following file and directory structure:


${C_PROJ_TPL_CAD}
blksld
liblist.txt file for logo sublib registration
<logoapplic><standard><language> directory for logo files
blklist.txt file for logo symbol name registration
libdata.txt file for sublibrary configuration
lib.loc
format
liblist.txt file for format library name registration
<application><standard><language> directory for format files
formlist.txt file for format name registration
<format>.scr
<format>.frm
plot directory not used in AdvaBuild, but must be available as dummy
liblist.txt

Example:

```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/c_dib_functions.sh

FormLibCreate -i gef iec en
```

The example above creates format directory ${C_PROJ_TPL_CAD}/difiecen/format/difiecen and logo directory ${C_PROJ_TPL_CAD}/difiecen/blksld/diliecen and the corresponding administrative files.

3.5.1.4 Updating Project-specific Format Libraries

To update project format libraries from external format libraries within the delivery structure, use shell function:

DibCopyFile [-f|-i] *<source pathname>*/*<filter> <destination pathname>*

The following items must be considered when copying format related files:

- the logo symbols must be copied explicitly to the logo directory
- the . dwg files for revision blocks must be copied together with .dwg files for formats.

Example: Creating format libraries and copying format definitions and logos.

```
#!/bin/ksh
. ${C_BIN_CAPE_SH}/dib_functions.sh

FormSourceDir=${C_STA_ES_LIB_CFORM}/gef/iec/en/for-mat/r2_1-0
LogoSourceDir=${C_STA_ES_LIB_CFORM}/gef/iec/en/-logos/r2_1-0
FormTargetDir=${C_PROJ_TPL_CAD}/gefiecen/format/gef-iecen
LogoTargetDir=${C_PROJ_TPL_CAD}/gefiecen/blksld/gel-iecen

FormLibCreate -i gef iec en

# Here is an example for DibCopyFile use. It copy all standard
# IEC Frames and Logos in project structure
# Think that you must add ’${C_PROJ_TPL_CAD}/gefiecen’ to
# variable ’@DIB.FORMAT_LIBS’ in all ’*.dib’ file before you
# can use these frames and logos with DIB or EDIB.
DibCopyFile -i ${FormSourceDir}/*.dwg ${FormTargetDir}
DibCopyFile -i ${LogoSourceDir}/*.dwg ${LogoTargetDir}
```

3.5.2 Configuring (Electrical) Diagram Builder

There are several files you can use to configure the behavior and setup of the (Electrical) Diagram Builder. The processing of the files is described in Section 3.1.3, Structure and Handling of Configuration Files.

You can make the following settings within the files:

- dib.sql : data base retrieval file on project level is used to configure selection of data from objects in the data base
- *<obj_type>*.dib: Object type specific setup file on project level is used to configure
  - acad path
  - symbol library path
  - frame library path
  - plot configuration file assignment
  - printer queue assignment
- dib.dmi: Title block mapping file on project level is used to configure the mapping of date selected within dib.sql file to attributes defined in title blocks.

The following predefined variables can be used within all configuration files on project level.

- DIB.OBJ_ID The object identification of the corresponding diagram object.
- DIB.OBJ_TYPE The object type of the corresponding diagram object.
- DIB.OBJ_TYPE_L The object type in small letters of the corresponding diagram object.
- DIB.FILE_NAME_PURE The pure file name of the underlying drawing file.
- DIB_FILE_NAME_PURE_L The pure file name in small letters of the underlying drawing file.
- DIB.ITEMDES The item designation part of the object identification.
- DIB.DODES The document designation part of the object identification.
- DIB.DKCODES The document kind classification code of the object identification.
- DIB.SHNOBLK The sheet number block part of the object identification.
- DIB.COMMAND The command that was used to invoke the Diagram Builder. Values are PRINT or OPEN.
- DIB.PRINT_MODE The print mode if the Diagram Builder was invoked by command PRINT. Values are PRINTER or FILE.
- DIB.PRINT_FILE_NAME_PURE The pure file name (no path prefix and no extension) of the print file if the Diagram Builder was invoked by command PRINT.
- DIB.PRINT_DIRECTORY The absolute path of the directory where the printed file should be placed to if the Diagram Builder was invoked by command PRINT and the mode was set to FILE.
- DIB.USER_NAME The AdvaBuild user name which was used to login to the Structure Builder.

You can use all environment variables that are set before starting the Structure Builder.

More information on how to use internal variables and environment variables is in Appendix D, Internal Variables and Environment Variables and in Section 3.4.3, Adaptation of title block mapping in Project.

3.5.2.1 Configuring data base Retrieval File

In the data base retrieval file dib.sql, SQL statements separated by an end can be defined in order to select data from the database, which can be used for title block mapping (see Section 3.5.2.3, Configuring Title Block Mapping File) or within the object type specific configuration file.

The following rules should be considered:

1. Each select statement should select only one row. Thus the WHERE-clause should describe a unique conditional expression.
2. Each item of a SELECT-clause must be separated by the expression “||’<!>’||,”
3. Each item of the SELECT -clause must be preceded by an expression of the form “‘<variablename>=’||”
4. Each select statement must be terminated by the word **end** which must be the only term within its line.

5. All text between an end and the next select will be ignored and can be used as comment or can be used for having conditional selections. See Section 3.4.3, Adaptation of title block mapping in Project for an example.

The following possibilities exist:

1. You can use internal variable expression anywhere. They can be used as placeholders as described in Figure 3-14 or can represent some other text parts of the select statement. Before the select statement is executed all those variable expression will be expanded. For more information on variable expressions syntax see Section D.2, Syntax Definition**.**
2. Multiple select statements can be formulated within this file. The select statements may retrieve data independently or may depend on prior select statements. In the later case, data selected by a statement is just within the where or from clause of an other select statement.
3. Values which were selected by the select statement can be assigned to internal variables by using the construct as described in rule 3 above. The name specified by “`’<name>=’`” will be the name of an internal variable if the select statement terminates successfully.

```
SELECT
’DIA.OBJ_ID=’||NVL(OBJ_ID,’’)||’<!>’
’DIA.OBJ_TYPE=’||NVL(OBJ_TYPE,’’)||’<!>’
’DIA.SYM_NAME=’||NVL(SYM_NAME,’’)||’<!>’
FROM OBJ_INSTANCE
WHERE OBJ_ID=@{DIB.OBJ_ID}
end

SELECT
’DIA.PREP_DATE=’||NVL(PREP_DATE,’’)||’<!>’
’DIA.PREP_NAME=’||NVL(PREP_NAME,’’)||’<!>’
’DIA.LANGUAGE=’||NVL(PREP_LANGUAGE,’’)||’<!>’
FROM @{DIA.OBJ_TYPE}
WHERE OBJ_ID=@{DIB.OBJ_ID}
end
```

Figure 3-14. Examples for syntax in configuration file dib.sql

The following table describes the internal variables which are selected by the delivered retrieval file. See Section 3.4.3, Adaptation of title block mapping in Project for a detailed description of the select statement.

Table 3-2. List of internal variables which are preset by DIB

Variable	Description
Attributes fetched from DSTRUC object	
DSTRUC.OBJ_ID	Object id of DSTRUC object which is currently related during PDB session.
DSTRUC.OWNER	Owner of document package, which will be currently generated and is identified by PDB.OBJ_ID.
DSTRUC.CUST_DIS	Attribute for free customer disposition which is defined in related DSTRUC object of the current document package

Table 3-2. List of internal variables which are preset by DIB (Continued)

Variable	Description
Attributes fetched from PROJECT object	
PROJECT.PROJ_NAME_1	Project name (description) 1 of the project
PROJECT.PROJ_NAME_2	Project name (description) 2 of the project
PROJECT.PROJ_NAME_3	Project name (description) 3 of the project
PROJECT.RESP_DEPT	The name of the responsible department for the project
PROJECT.CUSTOMER	The name of the customer
PROJECT.FACTORY	The name of the factory related to the project
PROJECT.OWNER	The name of the owner related to the project
PROJECT.PRO_NO	Project number
Attributes fetched from current DIAGRAM object	
DIA.OBJ_ID	The object id of the current diagram object
DIA.SYM_NAME	The symbolic name (comment) of the current diagram object
DIA.ITEM_DES_2	The item designation part 2 (NOTE: more details)
DIA.NO_OF_SH	Number of sheets of the diagram to which this sheet belongs
DIA.DOC_TYPE	The document type of the current diagram object
DIA.DOC_DES	The document designation of the current diagram object (NOTE: may be in conflict with document designation from OBJ_ID)
DIA.LANGUAGE	The used language
DIA.PREP_DATE	Preparation date of document
DIA.PREP_NAME	Name of preparing person
DIA.CHK_DATE	Check date of document
DIA.CHK_NAME	Name of checking person
DIA.APPR_DATE	Approve date of document
DIA.APPR_NAME	Name of approving person
DIA.BASED_ON	Reference to origin diagram, the diagram is based on
DIA.REPLACES	Reference to diagram, which is replaced by this diagram
DIA.REPL_BY	Reference to diagram, which replaces this diagram
DIA.LM_DATE_A	Last modified date, format: JJ-MM-DD
DIA.LM_DATE_B	Last modified date, format: DD.MM.JJ
DIA.LM_DATE_C	Last modified date, format: JJJJMMDD

Table 3-2. List of internal variables which are preset by DIB (Continued)

Variable	Description
Attributes fetched from parent object	
P.OBJ_ID	Object id of parent object
P.SYM_NAME	Comment of parent object
Attributes fetched from grand parent object	
PP.OBJ_ID	Object id of grand parent object
PP.SYM_NAME	Comment of grand parent object
Attributes fetched from great grand parent object	
PPP.OBJ_ID	Object id of great grand parent object
PPP.SYM_NAME	Comment of great grand parent object
Date fetched from system	
SYS.PR_DATE_A	Actual print date, format: JJ-MM-DD
SYS.PR_DATE_B	Actual print date, format: DD.MM.JJ
SYS.PR_DATE_C	Actual print date, format: JJJJMMDD
Attributes calculated by Diagram Builder functions (see Section 4.4.3.1)	
DIB.ITEMDES	The item designation extraction of DIA.OBJ_ID including *,=,==,+,++,- blocks
DIB.DODES	The item designation extraction of DIA.OBJ_ID including & blocks
DIB.DKCODE	The item designation extraction of DIA.OBJ_ID including Document Kind classification code block
DIB.SHNOBLK	The item designation extraction of DIA.OBJ_ID including Sheet number block

3.5.2.2 Configuring Object Type Specific Setup Files

The object type specific setup files are located on directory ${C_PROJ_MSC_DIB}/cust.

The following files exist:

- ci_dia.dib
- la_dia.dib
- pi_dia.dib
- hookup.dib
- drawing.dib.

Configuring Symbol Library Path

The symbol library path can be specified within the configuration file by use of the internal DIB variable DIB.SYMBOL_LIBS.

- Within the path definitions several path levels can be specified.
- Levels are separated by ';'.
- The definition must be written in one line.

Example:

```
@DIB.SYMBOL_LIBS =.....
'${C_PROJ_LIB_CAD}/cis@{STANDARD}@{LANGUAGE};....
 ${C_PROJ_LIB_CAD}/cis;.....
 ${C_PROJ_LIB_CAD}/drs'
```

NOTE

It can be distinguished only between two sublibraries with identical names within path definition. For circuit diagram symbols, only the first found sublibrary within path definition can be used.

Configuring Frame Library Path

The symbol library path can be specified within the configuration file by use of the internal DIB variable DIB.FORMAT_LIBS.

- Within the path definitions several path levels can be specified.
- Levels are separated by ';'.
- The definition must be written in one line.

Example:

```
@DIB.FORMAT_LIBS =.....
'${C_PROJ_TPL_CAD}/cif@{STANDARD}@{LANGUAGE};....
 ${C_PROJ_TPL_CAD}/cifiec@{LANGUAGE}'
```

Configuring ACAD path

Project specific directories containing drawings, symbols, lisp or font files can be defined here:

@ACAD = ''

This is necessary for the revision block insert function within (E)DIB. You must define the full path to the directory, where the format related .dwg files are located.

The definition of further paths may additionally be relevant when external drawings are imported, which are not made by the Diagram Builder and therefore use other definitions.

Configuring plotter files

For each valid format type a plotter shortname must be defined. The plotter shortname must refer to a directory below ${C_CAPE_ALL_MSC_DIB}/plt/plot, which includes the already configured plotter files:

```
@PLOTTER.A0L=’A4PS’
@PLOTTER.A1L=’A4PS’
@PLOTTER.A2L=’A4PS’
@PLOTTER.A3L=’A4PS’
@PLOTTER.A3P=’A4PS’
@PLOTTER.A4L=’A4PS’
@PLOTTER.A4P=’A4PS’
@PLOTTER.Q4P=’A4PS’
```

The Plant Documentation Builder supports only one queue at a time. Therefore, the plotter definition must be static and can not be set to specific format types.

```
@PLOTTER.PDB=’A4PS’
```

Within AdvaBuild the plotter with shortname ‘A4PS’ is predefined.

Configuring printer queues

For each format type a printer queue must be specified, to which the AutoCAD plot will be sent when print is performed via STB menu item **Object Print**. The value to be assigned is the name of an environment variable. Do not use the value of the environment variable by typing `@QUEUE.A1L=’$C_PRINT_POST’`.

```
@QUEUE.A0L=’C_PRINT_POST’
@QUEUE.A1L=’C_PRINT_POST’
@QUEUE.A2L=’C_PRINT_POST’
@QUEUE.A3L=’C_PRINT_POST’
@QUEUE.A3P=’C_PRINT_POST’
@QUEUE.A4L=’C_PRINT_POST’
@QUEUE.A4P=’C_PRINT_POST’
@QUEUE.Q4P=’C_PRINT_POST’
```

Configuring object type specific title block mapping

Internal variables can be defined here and assigned to values selected from the data base to specify, for example, an object type specific title block update.

3.5.2.3 Configuring Title Block Mapping File

The title block mapping is configured in file `${C_PROJ_MSC_DIB}/cust/dib.dmi.`

Use this file to describe the mapping of information from the data base to attributes within the title block. You can do this by assigning variables defined in `dib.sql` file or also in *<objecttype>*`.dib` file to variables which represent title block attributes.

Each mapping definition consists of two lines:

- The first line defines the attribute name within the title block.
- The second line describes the value respectively variable to be assigned to the attribute.

Example:

LANGUAGE
@{DIA.LANGUAGE:-}

NOTE

Appending '`:-`' to the variable means to generate an empty string, if the variable is not defined.

The list of relevant title block attributes is described in Section A.2, Standard Title Block Attributes or can be found in Section 3.4.3, Adaptation of title block mapping in Project.

Some internal variables can be used which contain certain designation blocks extracted from the object identification of the diagram object according to the naming rules taken from the IEC standard.

Those variables are:

- DIB.SHNOBLK
- DIB.ITEMDES
- DIB.DKCODE
- DIB.DODES.

For further description see Section 4.4.3.1, Inserting Diagram Objects in AdvaBuild.
An example for the mapping file as it is delivered in the standard configuration is included in Section 3.4.3, Adaptation of title block mapping in Project.

3.5.3 Configuring Plotter Settings

With your installation of the (Electrical) Diagram Builder the following standard plotter setting is included:

The standard plotter settings directory is

${C_CAPE_ALL_MSC_DIB}/plt/plot/a4ps

where

${C_CAPE_ALL_MSC_DIB}/plt

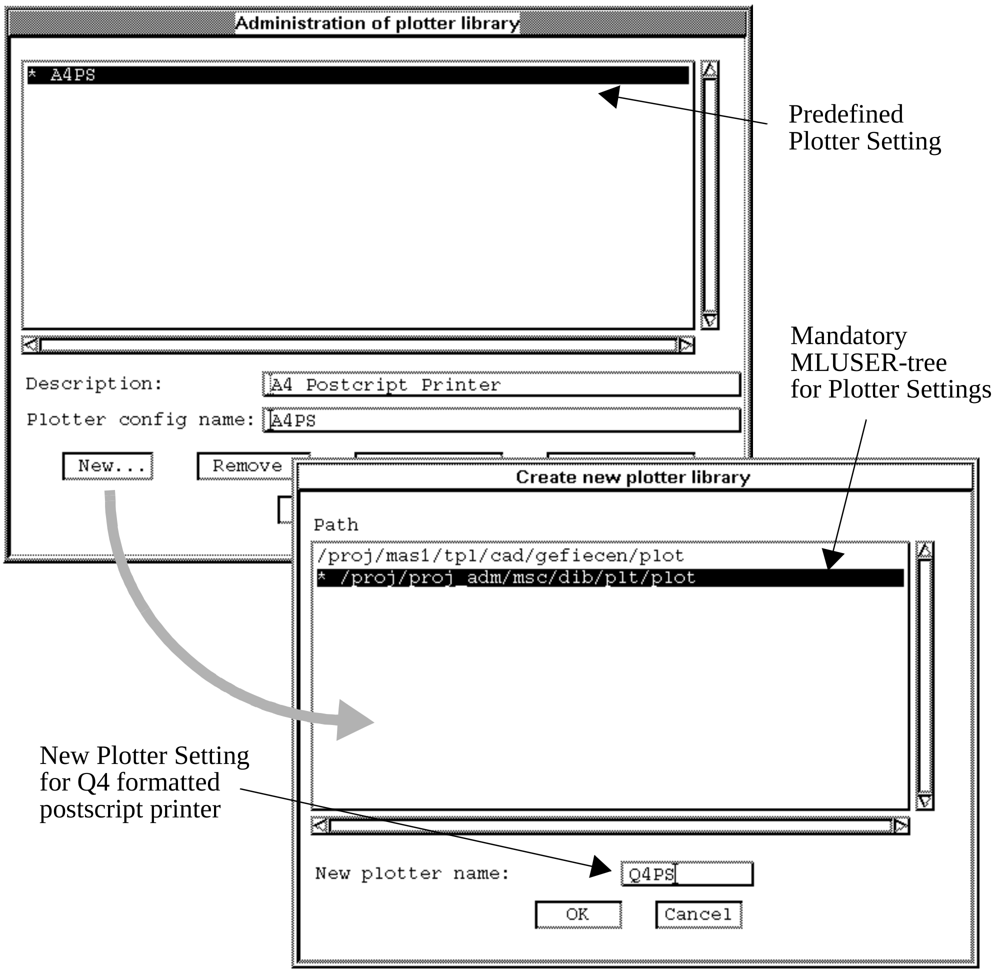
is the designated mluser tree for keeping plotter settings. All further settings must be placed within this mluser-tree.

Plotter setting directories can be created from within (E)DIB.

1. Choose **DIBPLOT** from the **DIB screen menu**. The **DIBPLOT screen menu** appears.
2. Choose **Edit.lib** from the **DIBPLOT screen menu**. The Administration of plotter library dialog box appears. The plotter setting directory a4ps (displayed A4PS) is shown.
3. You may include a new plotter setting with name Q4PS. Type in the New plotter name and click at the **OK** button. The respective directory will be created underneath

   ```
   ${C_CAPE_ALL_MSC_DIB}/plt/plot
   ```

   and the Administration of plotter library displays the additional settings name.
4. Click on the **OK** button within the Administration of plotter library dialog box.
5. Select **File Manager** from the Structure Builder's **Utility** menu.
6. Change to directory `${C_CAPE_ALL_MSC_DIB}plt` and start a new file manager from the **View New** menu within the current file manager window.
7. Change to directory a4ps within one file manager and change to directory q4ps within the other file manager.



* A4PS

Description: A4 Postcript Printer

Plotter config name: A4PS

Path

/proj/mas1/tpl/cad/gefiecen/plot
* /proj/proj_adm/msc/dib/plt/plot

New plotter name: Q4PS

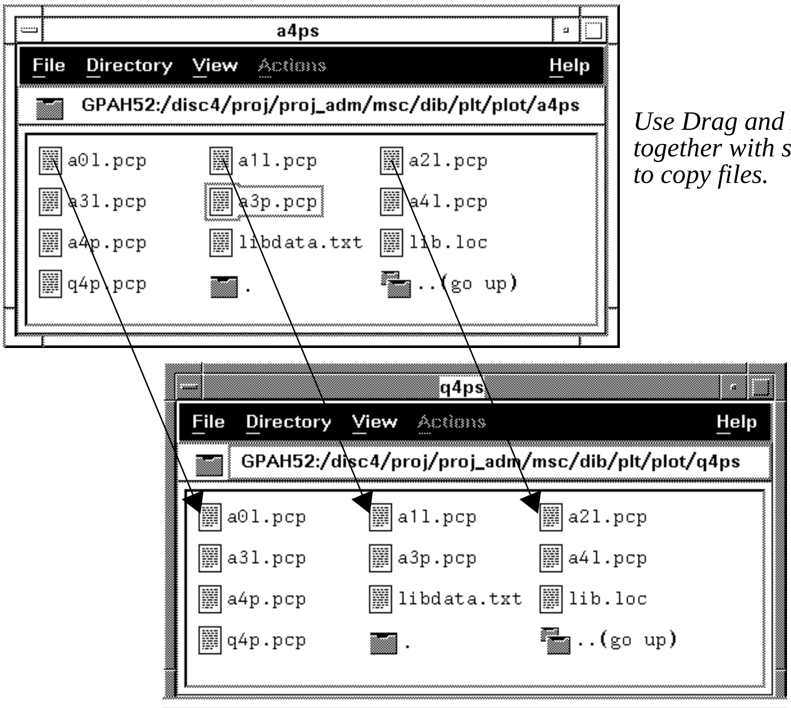
Predefined Plotter Setting

Mandatory MLUSER-tree for Plotter Settings

New Plotter Setting for Q4 formatted postscript printer

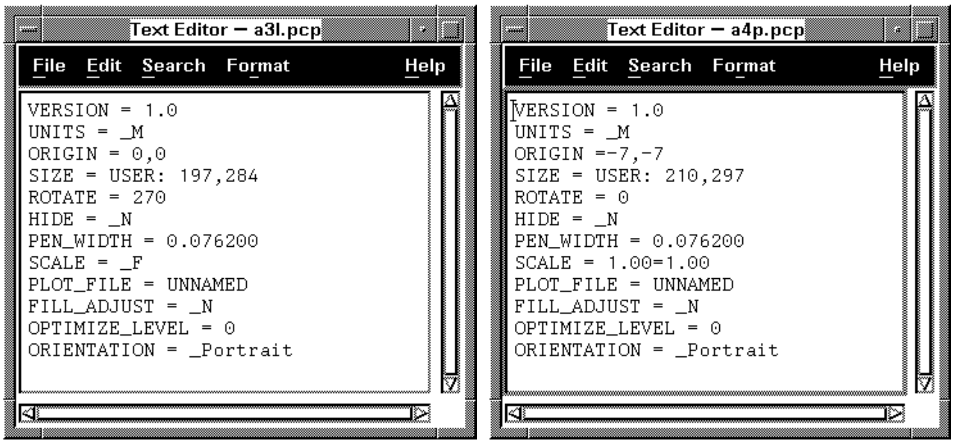
Figure 3-15. Administration of Plotter Library

8. Copy files *.pcp (nothing else) from the a4ps window to the q4ps window by using Drag & Drop while simultaneously pressing the **Shift** key.
9. Open files by double clicking at the respective icons within the q4ps window. The Text Editor window appears. (See examples below for files a3l.pcp and a4p.pcp)
10. Edit the files concerning the guidelines described below.



Use Drag and Drop together with shift to copy files.

Figure 3-16. File Copying



```
VERSION = 1.0
UNITS = _M
ORIGIN = 0,0
SIZE = USER: 197,284
ROTATE = 270
HIDE = _N
PEN_WIDTH = 0.076200
SCALE = _F
PLOT_FILE = UNNAMED
FILL_ADJUST = _N
OPTIMIZE_LEVEL = 0
ORIENTATION = _Portrait
```

```
VERSION = 1.0
UNITS = _M
ORIGIN =-7,-7
SIZE = USER: 210,297
ROTATE = 0
HIDE = _N
PEN_WIDTH = 0.076200
SCALE = 1.00=1.00
PLOT_FILE = UNNAMED
FILL_ADJUST = _N
OPTIMIZE_LEVEL = 0
ORIENTATION = _Portrait
```

Figure 3-17. AutoCAD pcp-files

The examples above are taken from the predefined plotter setting for A4 postscript printers. Meeting the guidelines described below will yield correct pcp-files.

NOTE

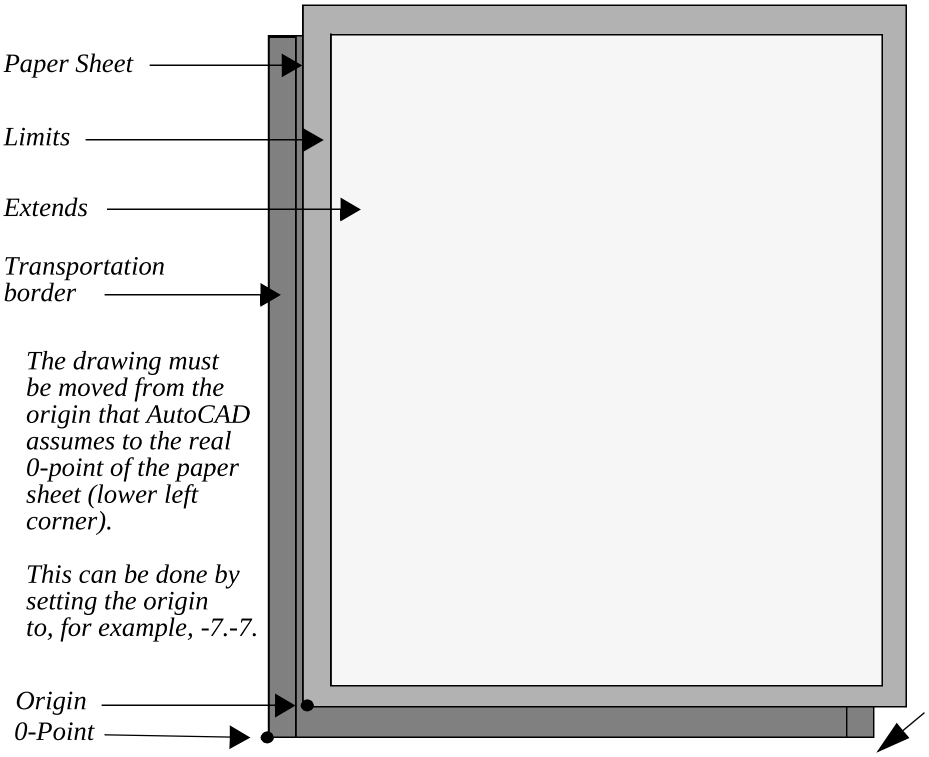
Printing from the Structure Builder and from within DIB is always based on the limits and never on the extends or special predefined windows. You must use the AutoCAD plot function if you want to do so.

- VERSION
  Must not be changed.
- UNITS
  Can be set to _M (millimeters) or _I (inches).
- ORIGIN
  In case of formats a0l, a1l, a2l, a3l, a3p the origin should be set to 0,0.
  In case of formats that should be printed 1:1 the origin must be shifted normally into native regions. (see below)
- SIZE
  In case of formats a0l, a1l, a2l, a3l, a3p the size should be set to what AutoCAD proposes (that is, 198,285 for A4 plotters) but must be changed slightly if AutoCAD's assumptions for the plotter are not exact (for example, one millimeter less - 197,284).
  In case of 1:1 plotting the size should be set to the same size defined for the A4P limits respectively Q4P limits. (that is, 210,297 respectively 210,279.4)
  Define

  SIZE = USER: 210,279.4

  in the case of Q4P.
- ROTATE
  Use 270 for landscape formats and 0 for portrait formats.
- HIDE
  Should be set to _N.
- PEN_WIDTH
  This setting is up to you.
- SCALE
  In case of formats a0l, a1l, a2l, a3l, a3p use _F (fit).
  In case of formats that should be printed 1:1 use 1.00=1.00
- PLOT_FILE
  Should be set to UNNAMED.
- FILL_ADJUST
  Should be set to _N.
- OPTIMIZE_LEVEL
  Should be set to 0.
- ORIENTATION
  Should be set to _Portrait.

For printing drawings 1:1 to the printer some unusual settings must be done. AutoCAD assumes that the printer (for example, A4 printer) has a border for transportation purposes and places the 0-point (origin of the paper) at location 7,7. As the Diagram Builder always prints out the whole limits this would leave a shifted printout in case of 1:1 plotting of A4 or Q4 formats. The picture below illustrates the action that will be forced by the recommended settings. As a result the transportation border will touch the limits but not the extends of the drawing. (However this depends on the plotter and its transportation mechanism.)



The drawing must be moved from the origin that AutoCAD assumes to the real 0-point of the paper sheet (lower left corner).

This can be done by setting the origin to, for example, -7.-7.

Figure 3-18. Changing the plotter origin

If the new plotter setting should become the current one, the entries within files <object type>.dib within directory ${C_PROJ_MSC_DIB}/cust must be updated. For more information on that see Section 3.5.2.2, Configuring Object Type Specific Setup Files**.**

In addition there must be a plotter defined within AutoCAD having the same short name as the name of the new plotter setting (for example, Q4PS). For more information see Section 3.5.4.2, Configuring Plotters.

3.5.4 Configuring AutoCAD

In the AdvaBuild environment two configuration files exist, and they reside on directory:

- ${C_CAPE_ALL_CFG_DIB}/mouse
  which is the valid configuration directory for use on X-Terminals
- ${C_CAPE_ALL_CFG_DIB}/digit
  which is the valid configuration directory for use on the Console.

These files exist to prevent AutoCAD from failing during start-up on X-Terminals.

All devices AutoCAD should have access to must be configured and registered in the current configuration file.

CAUTION

> As there are two different configuration files, one for use on X-Terminals and one for use on the Console, all configuration changes which will be done from X-Terminals do not affect the configuration file for the Console and vice versa.

You can reconfigure AutoCAD during a normal open call to AutoCAD. Be aware that the changes are done for the whole project and are not user specific!

At this time, a variety of configurations can be set respectively changed. For further information, see the *AutoCAD Release 12, Interface, Installation and Performance Guide*, Chapter 3, "Configuring AutoCAD."

3.5.4.1 Configuring Operating Parameters

To avoid problems after importing dwg files, font files, etc., from other platforms such as DOS, configure the following.

1. Choose item **7. Configure operating parameters** from the AutoCAD **Configure** menu. A submenu is displayed.
2. Choose item **8. File name case** from the current submenu. The following question is displayed:

   Do you want to force all file names forced to lower case? <N>

3. Enter `N`.
4. Choose item **5. Placement of Temporary Files** from the current submenu. The following question is displayed:

   Enter directory name for temporary files, or DRAWING to
   place them in the same directory as the drawing being edited.
   <DRAWING>:

5. Enter `/tmp`.

NOTE

> Although AutoCAD offers a variety of options to change parameters to meet various needs and user preferences, it is advisable that you use only the recommended options to avoid problems.

Other operating parameters:

- Alarm on Error: Can be defined in any way you choose.
- Initial drawing setup: Do not change
- Default Plot File Name: Do not change.
- Plot Spooler Directory: Do not change.
- Network Node Name: Do not change.

- Network Node Name: Do not change.
- Automatic - Save Feature: Can be set any way you choose.
- Full-Time CRC Validation: Can be set any way you choose.
- Automatic Audit after IGESIN, DXFIN, or DXBIN: Can be set any way you choose.
- Select Release 11 hidden line removal algorithm: Can be set any way you choose.
- Login Name: Do not change.
- File Locking: Do not change.
- License Server: Do not change.

3.5.4.2 Configuring Plotters

To configure your plotter, give the selected plotter name as described below.

1. Choose item **5**. **Configure Plotter** from the AutoCAD **Configure** menu. The current plotter name appears. For example:

   Your current plotter is: None

   At the same time the **Plotter Configuration** Menu appears.
2. If the current plotter is set to none choose item **1. Add a plotter configuration**.
3. A list of available plotters is displayed, followed by this prompt:

   Select device number or ? to repeat list <1>:

   Enter the number corresponding to the *PostScript device* and press <**Return**>.
4. Enter the number corresponding to the plot resolution you prefer and press <**Return**>.
5. AutoCAD prompts

   Do you want color output? <N>

   Press <**Return**>.
6. AutoCAD prompts

   Do you wish to append a ^Z? <N>

   Press <**Return**>.
7. AutoCAD prompts

   Do you wish to change the serial port? <N>

   Press <**Return**>.
8. AutoCAD prompts for the plotter device name.

   Enter serial device name for plotter or . for none </dev/tty00>:

   Enter **.** to select no device name.

9. AutoCAD may prompt now the following list of settings

Plot optimization level = 1

Plot will NOT be written to a selected file

Sizes are in Inches and the style is portrait

Plot origin is at (0.00,0.00)

Plotting area is 8.00 wide by 10.50 high (A size)

Plot is NOT rotated

Area fill will NOT be adjusted for pen width

Hidden lines will NOT be removed

Plot will be scaled to fit available area

Do you want to change anything? (No/Yes/File) <N>:

You may change all or none of these settings.

10. If you choose **N** AutoCAD prompts for the plotter description

    Enter a description for this plotter:

    Enter **A4PS**.

NOTE

This holds for the standard plotter configuration. You may also define another name (for example, Q4PS). The naming must follow the convention <format><printerlanguage>, where format, for instance Q4 or A4, and language might be ps for postscript or hpgl and so on. Make the required changes as described in Section 3.5.3, Configuring Plotter Settings and Section 3.5.2.2, Configuring Object Type Specific Setup Files.

11. You are now back in the **Plotter Configuration** Menu and may leave the plotter configuration.

3.5.4.3 Configuring Video Display

Configure your video display if not already done.

1. Choose item **3. Video Display** from the AutoCAD **Configure** menu. The current display name appears. For example:

    Your current video display is: Motif Display - by Autodesk
    Do you want to select a different one? <N>

    Enter **Y**.

2. AutoCAD searches for the available video display, displays a list, and prompts:

    Select device number or ? to repeat list <1>:

    Enter **3** to select the **Motif/Xlib Display.**

3. AutoCAD prompts:

   If you have previously measured the height and width of a "square" on your graphics screen, you may use these measurements to correct the aspect ratio.

   Would you like to do so? <N>

   Enter **Y**.

4. AutoCAD prompts:

   Do you want a status line? <Y>

   Enter **Y**.

5. AutoCAD prompts:

   Do you want a command prompt area? <Y>

   Enter **Y**.

6. AutoCAD prompts:

   Do you want a screen menu area? <Y>

   Enter **Y**.

7. You are now back in the AutoCAD configuration menu and may leave the configuration by entering **`0`**.

3.6 Configuration Menus

This section is not applicable.

Chapter 4 Operation

4.1 Product Operation

The (Electrical) Diagram Builder is made up of the following general modes/tasks:

- Configuration of AutoCAD—described in Section 3.5.4, Configuring AutoCAD
- Configuration of the Diagram Builder—described in Section 3.5.2, Configuring (Electrical) Diagram Builder
- Installation of Format/Symbol Libraries—described in Section 3.4.2, Installing Symbol and Format Libraries in Project
- Building symbols—described in Section 4.3.2, Building Symbol Blocks and Section 4.3.3, Building Composed Symbol Blocks
- Building Diagrams—described in Section 4.3.1, Building Diagrams.

NOTE

Before starting the (Electrical) Diagram Builder the first time see Section 3.3, Application Start-up.

4.2 Operation Overview

4.2.1 General Mechanisms

Use the Diagram Builder to edit the graphics in AdvaBuild diagrams. Every diagram is represented in the AdvaBuild data base by a related diagram object instance. Diagram Builder object types are:

- **DRAWING**

  Use for drawings such as overview and block diagrams and other miscellaneous drawings that do not fall into any of the predefined categories below.

- **PI_DIA**

  Use for pipe and instrumentation diagrams within the function-oriented plant structure.

- **CI_DIA**

  Use for electrical circuit diagrams.

- **LA_DIA**

  Use for layout diagrams (for example, plant layout, operator room layout, cubicle layout, rack layout).

- **HOOKUP**

  Use for hookup diagrams related to sensors and actuators for instrumentation and control.

Apply normal AdvaBuild operating methods to the above objects. For example:

- **Insert**

  Use **Insert** to create a diagram object instance and place it into the tree structure or in list representation.

- **Update**

  Use **Update** to specify the relevant attributes of a diagram object instance.

- **Delete**

  Use **Delete** to remove a diagram object instance from the tree structure and remove the associated drawing's contents.

- **Move**

  Use **Move** to take a diagram object instance to a new location.

- **Rename**

  Use **Rename** to change the name of a diagram object instance (does not affect the UNIX representation of the associated drawing).

- **Copy**

  Use **Copy** to reproduce a diagram object instance and the associated drawing's content. Only the data base representation of the associated drawing will be copied. If you **Copy** using the Template Builder, the drawing's contents will not be copied.

- **Open/Open Type for editing purposes**

  You can **Open** in Diagram view as well as Template view.

  Open Type can only be used in Template Builder.

- **Print for documentation purposes.**

  Use **Print** to print a single object within Structure Builder.

Most title block attributes in the diagram formats are updated from the data base when:

- Opening diagram objects via **STB Open View Diagram**
- Printing diagram objects via **STB Print View Diagram** or via Plant Documentation Builder.

Other objects and attributes defined within a diagram's graphics are not connected to the AdvaBuild database, so there is no automatic update of information between symbols in diagrams (drawings) and objects (instances) in the AdvaBuild database.

Use predefined symbol blocks to build the graphical content of diagrams. Predefined symbol blocks are administered in libraries specific for each diagram object type. Symbol and format libraries are handled specifically for each project.

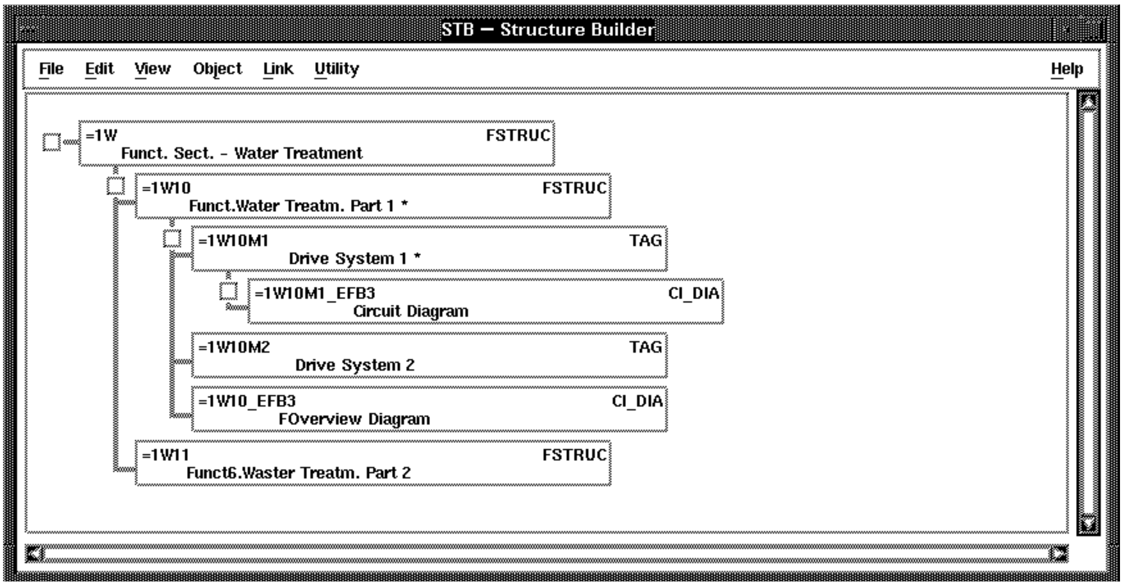


Figure 4-1. Structure Builder Main Window with Tree Structure

4.2.2 Privileges

To create and edit diagrams, you must have at least write privileges within the project. With write privileges, you can also make new symbol sublibraries directly within the (Electrical) Diagram Builder.

To enter the Diagram Builder configuration via STB menu **Utility Configure Diagram Builder** you must have configure privileges.

4.2.3 Locking Mechanisms

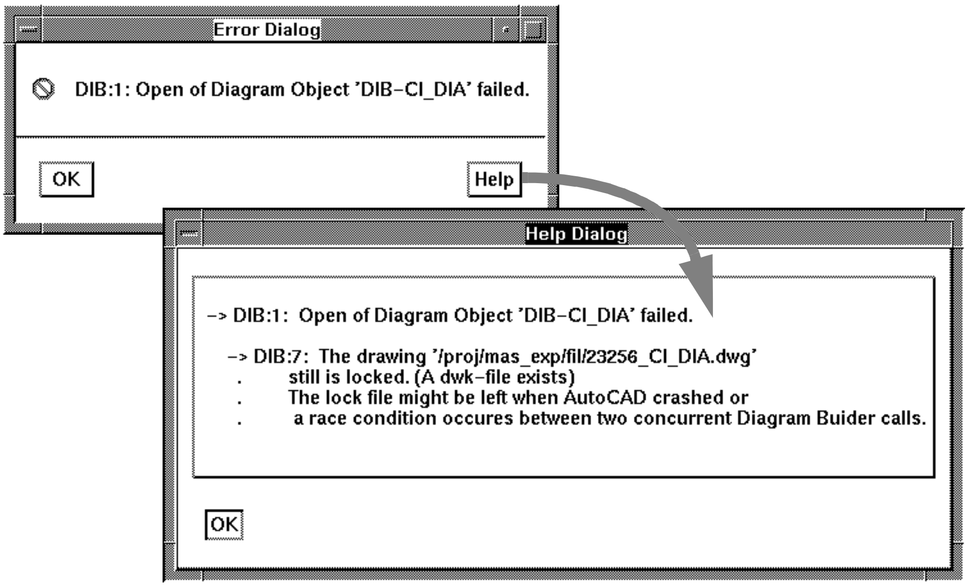
Locking mechanism exists:

- on the diagram object level
- on the drawing level
- on the symbol sublibrary level
- on the format sublibrary level.

Locking on **object level** makes sure, that only one user can work on the graphical content of one diagram object at the same time. If the same diagram object is opened with view Diagram a second time (by same or other user), a temporary copy of the .dwg file is created on the file system with write protection. The Diagram Builder is started with read only access rights. In addition it is not possible to change the object's data via the Template Builder.

For each drawing, AutoCAD sets a write lock (dwk-file) to stop multiple users from working on the same drawing file at the same time. It may happen that AutoCAD aborts because of internal errors. Then the current locking file (dwk-file) of the current drawing will leaf at the file system (that is, on directory .../proj/*<project short name>*/fil). This will be reported by the (Electrical) Diagram Builder by displaying the following Error Dialog / Help Dialog boxes.

In the mentioned case the lock file must be removed manually. The lock file has the same name as the drawing but has the extension .lck.



DIB:1: Open of Diagram Object 'DIB-CI_DIA' failed.

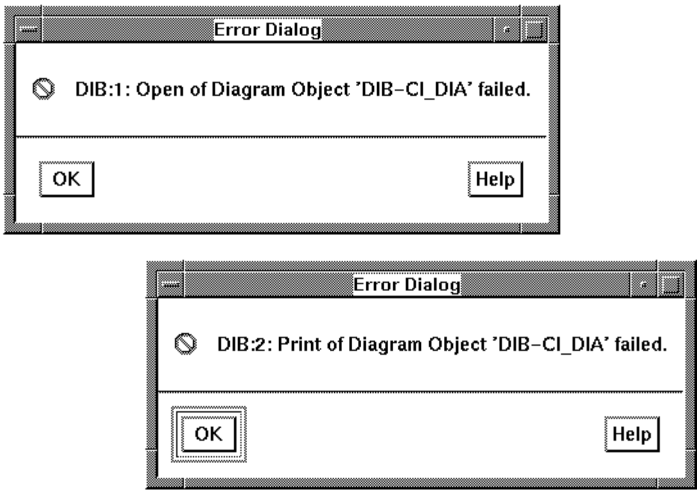
```
-> DIB:1:  Open of Diagram Object 'DIB-CI_DIA' failed.

   -> DIB:7:  The drawing '/proj/mas_exp/fil/23256_CI_DIA.dwg'
    .        still is locked. (A dwk-file exists)
    .        The lock file might be left when AutoCAD crashed or
    .         a race condition occures between two concurrent Diagram Buider calls.
```

Figure 4-2. Error dialog

Adverse conditions may exist if a drawing is opened twice for read only or printed twice at the same time. During start-up of AutoCAD a lockfile exists which will be reset afterwards. However a second open for read only or print method tries to lock the same file within this short time period. In these cases a second start of AutoCAD will solve the problem.

Locking on the symbol sublibrary level and format sublibrary level (within the Diagram Builder) makes sure, that symbol libraries on sublibrary level are protected from being updated by different persons in parallel. The mechanisms work on the file level. If editing/updating symbol libraries is not properly finished, sublibraries may remain locked. For those cases, use the **Unlock** button of the Administration of symbol libraries or the Administration of form setup dialog boxes within the Diagram Builder to unlock resources.

4.2.4 Failure Mechanisms

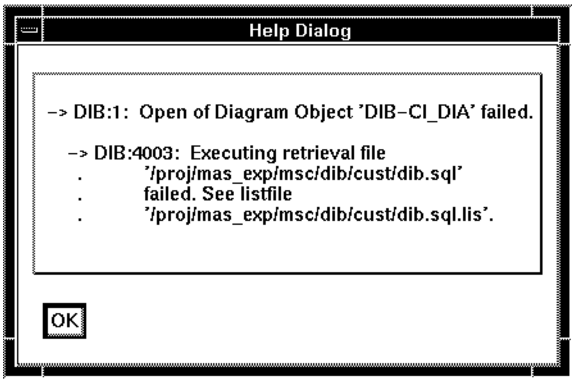


DIB:1: Open of Diagram Object 'DIB–CI_DIA' failed.

DIB:2: Print of Diagram Object 'DIB–CI_DIA' failed.

Figure 4-3. Failure Mechanisms Error Dialog Box

Whenever an error occurs during the start-up of the (Electrical) Diagram Builder, the Error Dialog box depicted in Figure 4-3 is shown. This holds for the print and open method. Then a Help Dialog box is displayed as shown in Figure 4-2 and in Figure 4-4. The Help Dialog box can be terminated when clicking the **OK** button. For leaving the Error Dialog box, click the **OK** button of the respective box.



```
-> DIB:1:  Open of Diagram Object 'DIB–CI_DIA' failed.

  -> DIB:4003:  Executing retrieval file
   .          '/proj/mas_exp/msc/dib/cust/dib.sql'
   .          failed. See listfile
   .          '/proj/mas_exp/msc/dib/cust/dib.sql.lis'.
```

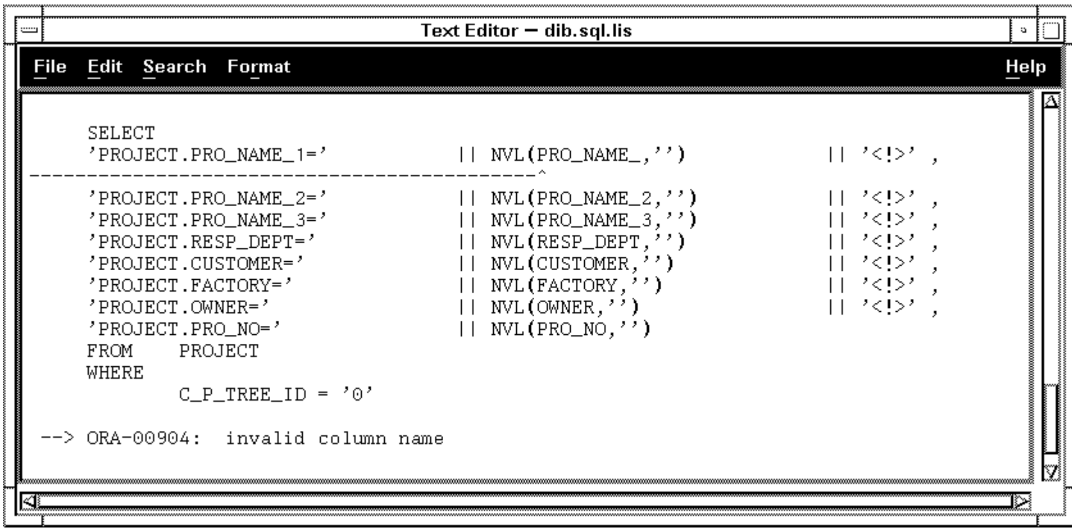
Figure 4-4. Help Dialog box

Whenever syntax errors occur within files:

- ci_cia.dib, la_dia.dib, pi_dia.dib, drawing.dib, hookup.dib
- dib.sql
- dib.dmi
- pre.dib, post.dib (on directory .../proj/proj_adm/msc/dib/cust).

This will be mentioned within the Help Dialog box as shown in Figure 4-4. For these files a list file will be created having the same name as the original file plus an extra extension **lis**.

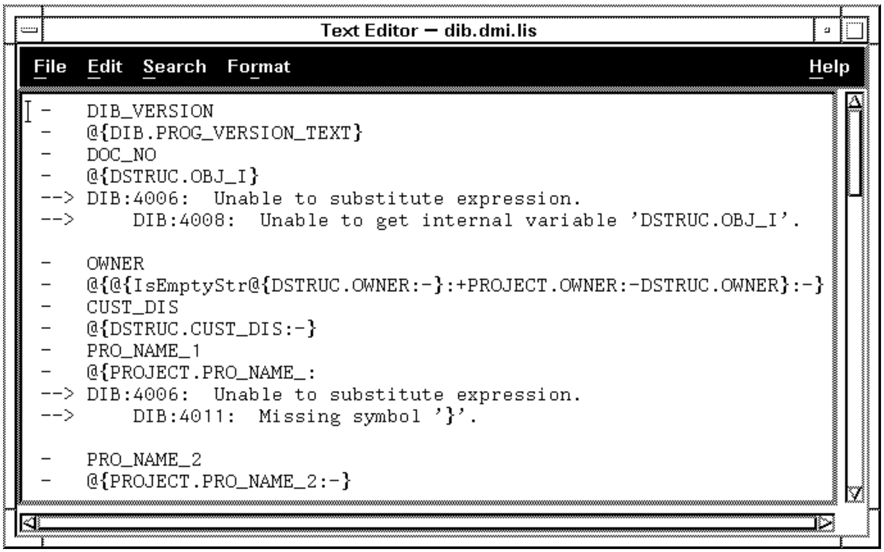
Figure 4-5 shows how an error within dib.sql will be shown. The error position will be marked by an arc. The corresponding error message is displayed underneath the sql statement.



```
     SELECT
     'PROJECT.PRO_NAME_1='           || NVL(PRO_NAME_,'')            || '<!>' ,
--------------------------------------------^
     'PROJECT.PRO_NAME_2='           || NVL(PRO_NAME_2,'')           || '<!>' ,
     'PROJECT.PRO_NAME_3='           || NVL(PRO_NAME_3,'')           || '<!>' ,
     'PROJECT.RESP_DEPT='            || NVL(RESP_DEPT,'')            || '<!>' ,
     'PROJECT.CUSTOMER='             || NVL(CUSTOMER,'')             || '<!>' ,
     'PROJECT.FACTORY='              || NVL(FACTORY,'')              || '<!>' ,
     'PROJECT.OWNER='                || NVL(OWNER,'')                || '<!>' ,
     'PROJECT.PRO_NO='               || NVL(PRO_NO,'')
     FROM    PROJECT
     WHERE
             C_P_TREE_ID = '0'

 --> ORA-00904:  invalid column name
```

Figure 4-5. List file for dib.sql

Figure 4-6 shows how errors within dib.dmi will be shown. Text lines are preceded by a minus '-' and lines describing the errors are preceded with an arc '-->'.



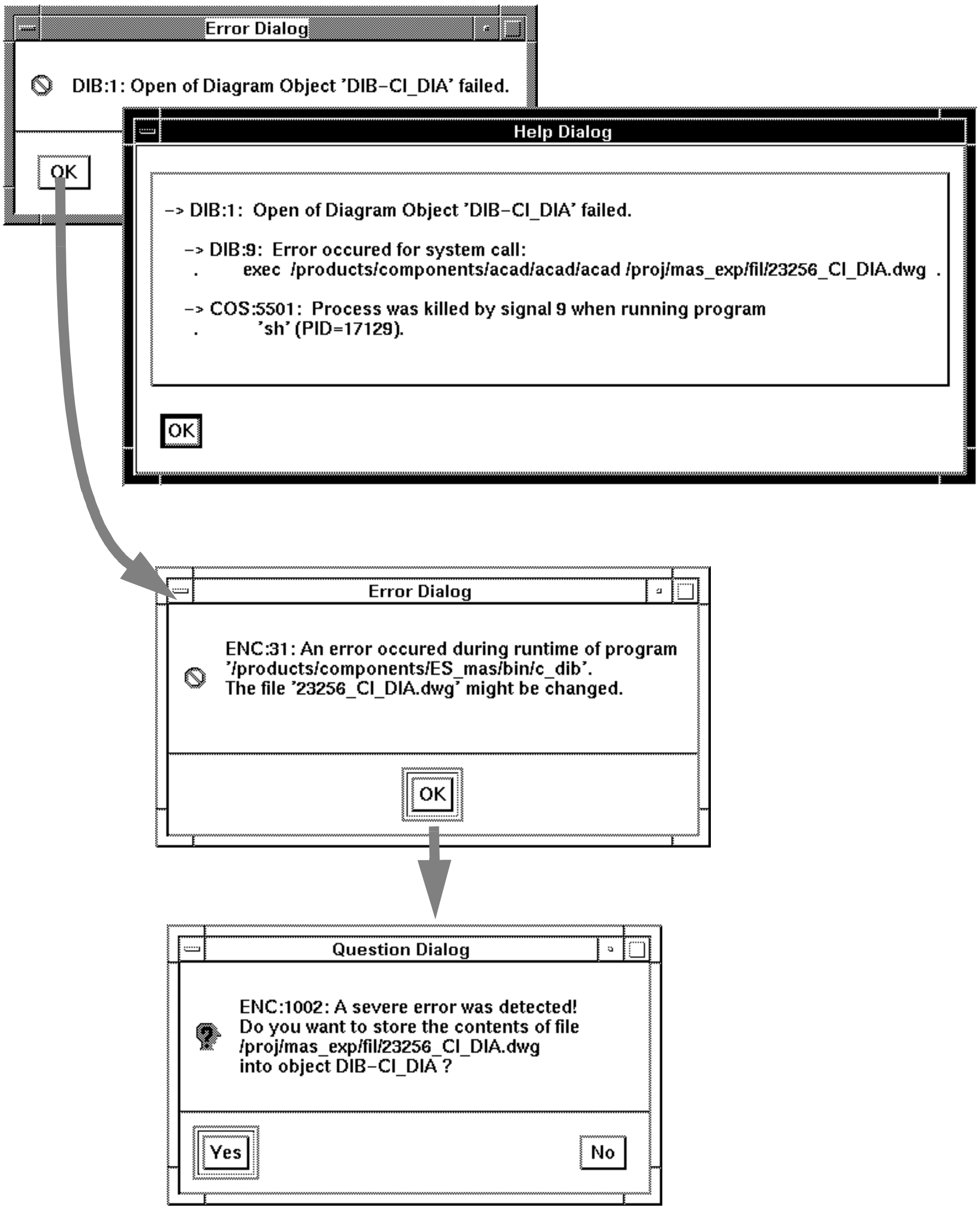
```
 -   DIB_VERSION
 -   @{DIB.PROG_VERSION_TEXT}
 -   DOC_NO
 -   @{DSTRUC.OBJ_I}
 --> DIB:4006:  Unable to substitute expression.
 -->     DIB:4008:  Unable to get internal variable 'DSTRUC.OBJ_I'.

 -   OWNER
 -   @{@{IsEmptyStr@{DSTRUC.OWNER:-}:+PROJECT.OWNER:-DSTRUC.OWNER}:-}
 -   CUST_DIS
 -   @{DSTRUC.CUST_DIS:-}
 -   PRO_NAME_1
 -   @{PROJECT.PRO_NAME_:
 --> DIB:4006:  Unable to substitute expression.
 -->     DIB:4011:  Missing symbol '}'.

 -   PRO_NAME_2
 -   @{PROJECT.PRO_NAME_2:-}
```

Figure 4-6. List file for dib.dmi

Errors that occur within files with extension dib will be represented like those in dib.dmi.

An error might occur before starting the (Electrical) Diagram Builder. Then the error message is different from those described in Figure 4-3.

Sometimes a Question Dialog box is displayed as described in Figure 4-7.



DIB:1: Open of Diagram Object ’DIB-CI_DIA’ failed.

```
-> DIB:1:  Open of Diagram Object ’DIB-CI_DIA’ failed.

   -> DIB:9:  Error occured for system call:
    .      exec  /products/components/acad/acad/acad /proj/mas_exp/fil/23256_CI_DIA.dwg  .

   -> COS:5501:  Process was killed by signal 9 when running program
    .          ’sh’ (PID=17129).
```

ENC:31: An error occured during runtime of program
’/products/components/ES_mas/bin/c_dib’.
The file ’23256_CI_DIA.dwg’ might be changed.

ENC:1002: A severe error was detected!
Do you want to store the contents of file
/proj/mas_exp/fil/23256_CI_DIA.dwg
into object DIB-CI_DIA ?

Figure 4-7. Error scenario

The described situation represents a severe error where the (Electrical) Diagram Builder cannot determine if the drawing contents should be stored back to the database.

When opening the same object again with view DIAGRAM the situation described within Figure 4-2 may occur.

4.3 Operation Tutorial

This tutorial provides a quick hands-on introduction on how to:

- handle diagram objects in the data base.
- edit the graphical content of diagram objects
- build new symbols
- build new composed symbols
- print diagrams
- adapt Diagram Builder configuration.

NOTE

Before starting the (Electrical) Diagram Builder the first time see Section 3.3, Application Start-up.

4.3.1 Building Diagrams

4.3.1.1 General Aspects

Diagram file handling

The data base views diagrams as file objects. The diagram files (.dwg) are available on the directory ..../proj/*<project>*/fil. After modification the file is stored in the database, but it remains on the file system.

During next access, the time stamp of the file representation and data base representation is compared. If the timestamp is identical, the file system representation is used. If the representation on filesystem is not available, it is extracted out of the database.

Library setup

When using diagram format and symbol libraries, the correct setup of the attributes STANDARD and LANGUAGE is important.

Before starting to build diagrams, make sure that:

- a format library is installed in project (normally the same for all types of diagrams) (below directory ..../proj/*<project>*/tpl/cad)
- the symbol libraries are installed in project (specific to diagram object type) (below directory ..../proj/*<project>*/lib/cad).

4.3.1.2 STB/TEB Operations

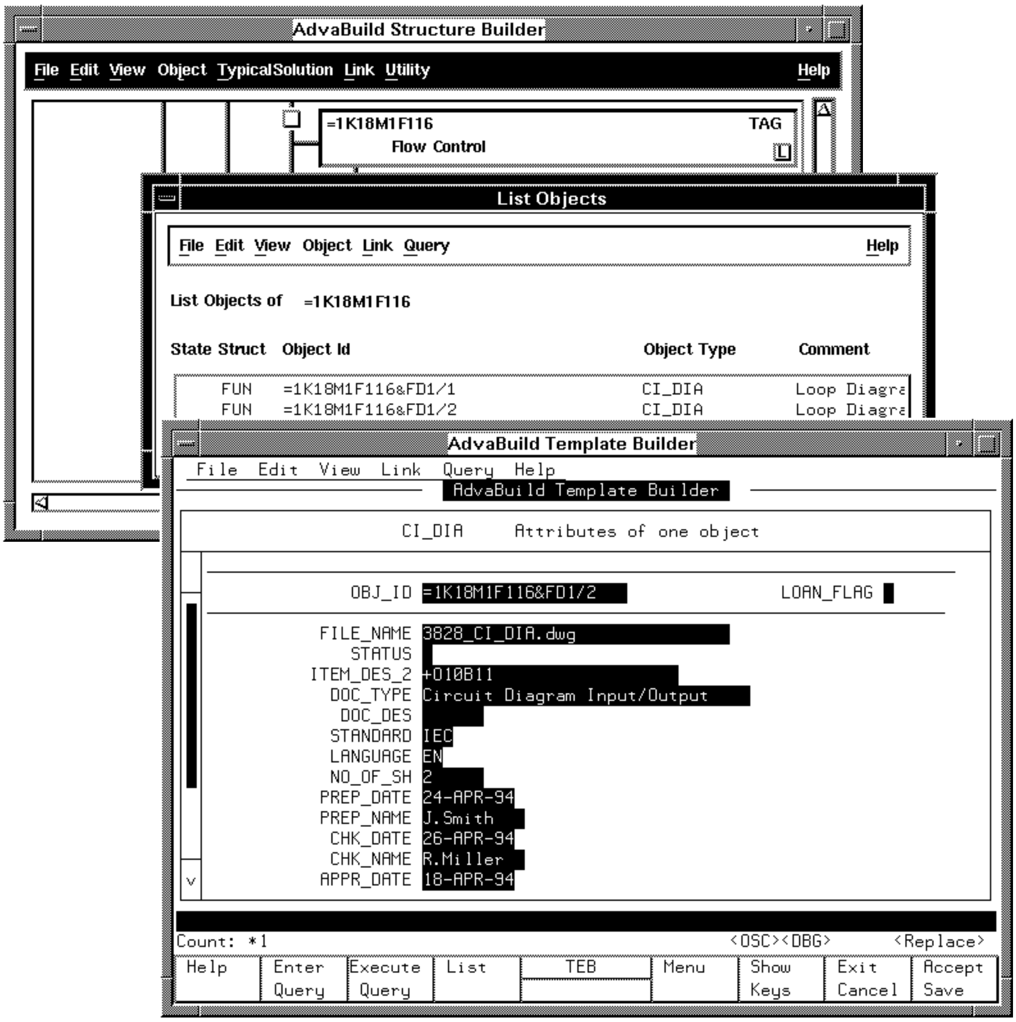
You can insert diagram objects via STB or TEB or you can copy available diagram objects via STB. You must insert a diagram object for each sheet.

NOTE

> If you insert diagram objects via TEB, use the TEB menu item **Edit Insert** at least for the first record!

It is recommended that you handle diagrams as list objects within STB. For the object id of the diagram object, follow the naming rules described in Section 4.4.3.1, Inserting Diagram Objects in AdvaBuild. In this case you can have certain designation blocks within the object id displayed in the title block automatically.

The next step is to edit the diagram related attributes. Therefore open the Template view on the diagram object and enter values.


AdvaBuild Structure Builder
File Edit View Object TypicalSolution Link Utility Help
=1K18M1F116 TAG
Flow Control
List Objects
File Edit View Object Link Query Help
AdvaBuild Template Builder
File Edit View Link Query Help
AdvaBuild Template Builder
Help
Enter Query
Execute Query
List
TEB
Menu
Show Keys
Exit Cancel
Accept Save

List Objects of =1K18M1F116

State	Struct	Object Id	Object Type	Comment
	FUN	=1K18M1F116&FD1/1	CI_DIA	Loop Diagra
	FUN	=1K18M1F116&FD1/2	CI_DIA	Loop Diagra

CI_DIA Attributes of one object

OBJ_ID =1K18M1F116&FD1/2 LOAN_FLAG

FILE_NAME 3828_CI_DIA.dwg
STATUS
ITEM_DES_2 +010B11
DOC_TYPE Circuit Diagram Input/Output
DOC_DES
STANDARD IEC
LANGUAGE EN
NO_OF_SH 2
PREP_DATE 24-APR-94
PREP_NAME J.Smith
CHK_DATE 26-APR-94
CHK_NAME R.Miller
APPR_DATE 18-APR-94

Count: *1 <OSC><DBG> <Replace>

Figure 4-8. Related Attribute Editing

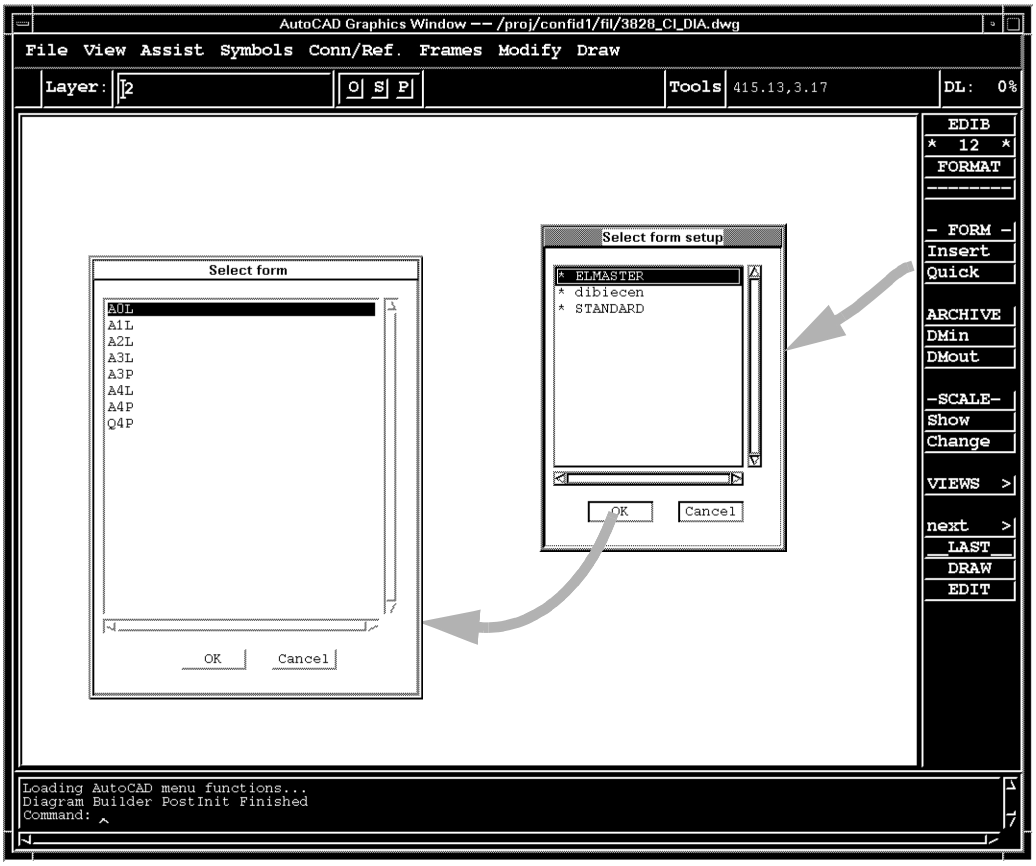
NOTE

It is not possible to update the attributes STANDARD and LANGUAGE afterwards.

4.3.1.3 Common DIB/EDIB Operations

The following steps are common for DIB and (E)DIB.

1. Setup for scale.
   For diagrams of type LA_DIA, HOOKUP and DRAWING, first select a scale factor from the right hand side (E)DIB screen menu. This scale factor will be used to scale the format when it is inserted.
2. Insert an empty format
   a. Choose **FORMAT, FORM Insert** from (E)DIB screen menu.
   b. Choose format sublibrary within appearing **Select form setup** selector box (this is relevant only if several format libraries exist in parallel).
   c. Choose format from appearing **Select form** selector box.



```
Loading AutoCAD menu functions...
Diagram Builder PostInit Finished
Command:
```

Figure 4-9. Inserting empty forms

Now the selected format including the frame and title block is inserted and the title block attributes are updated automatically from the data base.

3. Update title block manually
   To update the title block attributes manually you should choose **Modify Edit Attribute Value** from (E)DIB pull down menu (or enter just **ddatte** in (E)DIB command line) and pick title block by pointing device.

   Within the appearing **Edit attributes** dialog box you should only edit those attributes which are not updated from the database! Otherwise those are overwritten during next open.

   If you want to edit all title block attributes directly only in the diagram, you can switch off the update from the database. Therefore choose **DIBCONFIG CONFIG** from (E)DIB menu and switch button **Automatic updating From data base** off.

4. Insert revision blocks
   For revision handling within diagrams, revision blocks can be inserted into the diagram and the revision block attributes can be edited manually. Use the following procedure:

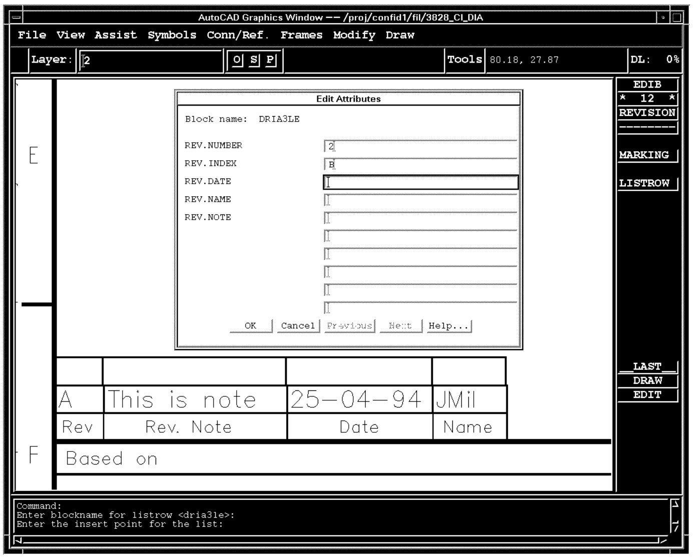
   a. Zoom to area where revision block should be placed (use, for example, **View Zoom Window** from (E)DIB pull down menu).

   b. Choose **REVISION LISTROW** from right hand side screen menu.

   c. Confirm proposed revision block name (or enter name of another one) within (E)DIB command area.

   d. Place revision block on top of revision footer block. If snap is active, switch snap temporarily off via S button. If placement was not correct, use AutoCAD command **move** for proper positioning.

   e. Edit revision block attributes.



Layer: 2

Block name: DRIA3LE

REV.NUMBER 2
REV.INDEX B
REV.DATE
REV.NAME
REV.NOTE

A	This is note	25–04–94	JMil

```
Command:
Enter blockname for listrow <dria3le>:
Enter the insert point for the list:
```

Figure 4-10. Revision Block Attribute Editing

4.3.1.4 DIB Operations

Inserting symbols from symbol library

If there are symbol libraries available for a certain application, you can select symbols via slide menus. Within DIB choose **SYMLIB Insert** screen menu, select sublibrary from list and select symbol via double click on icon. Within EDIB you can better use the **Symbols** pull down menu for symbols selection.

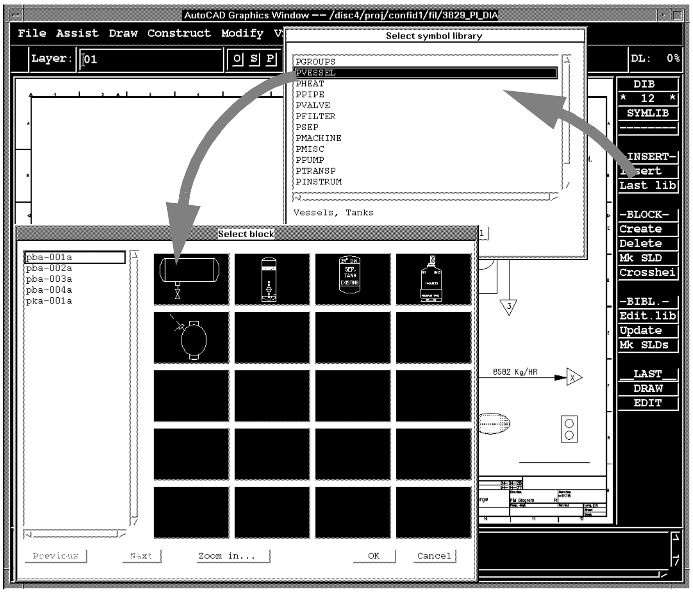


Figure 4-11. Symbol Insertion

Adding free graphics

To add pure free graphics like circles, rectangles, and lines, choose drawing functions from **DRAW** menu within DIB or use just AutoCAD commands.

4.3.1.5 EDIB Circuit Operations

NOTE

The EDIB Circuit specific functions can only be used in the described way if you have the option Electrical Diagram Builder Circuit available. Otherwise see Section 4.3.1.3, Common DIB/EDIB Operations for more information.

1. Insert format (see Section 4.3.1.3, Common DIB/EDIB Operations).
   Recommended format for general circuit diagrams is A3L. For loop diagrams also A4P or A3P may be used.
2. Editing the circuit diagram can be done, for example, in the following steps:

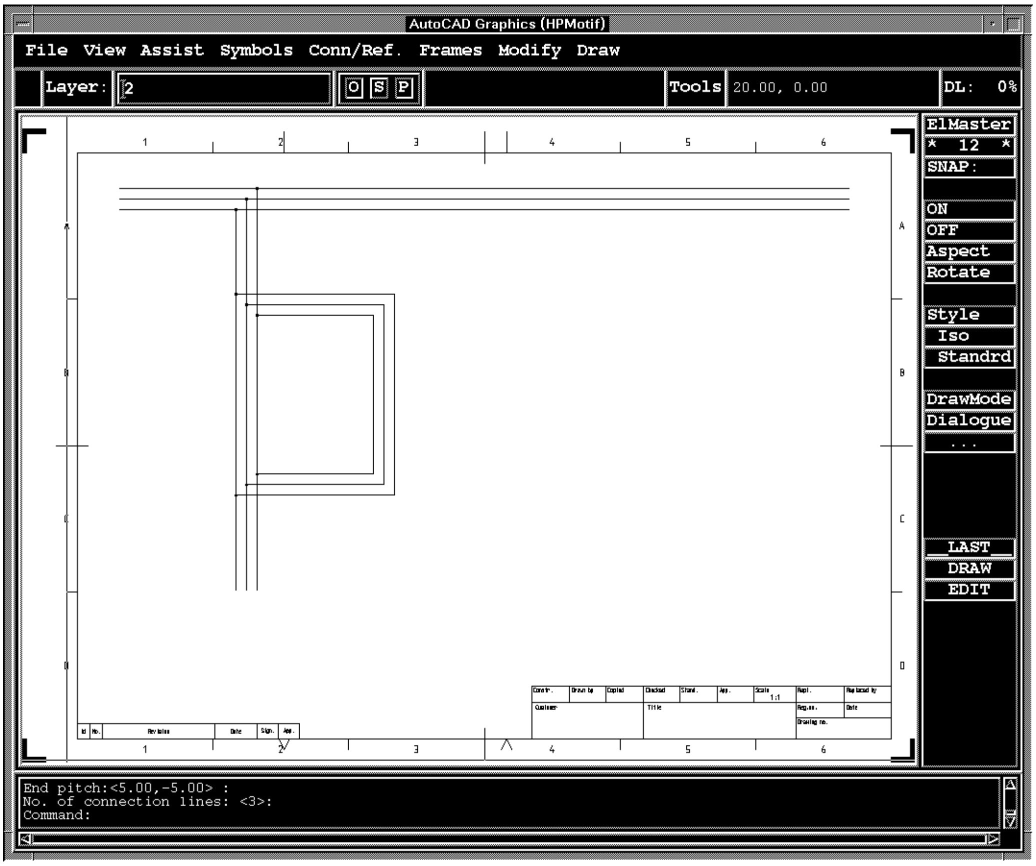
a. Check whether snap is active and set to 5 mm.
Otherwise set snap at first to 5 mm by entering **`snap 5`** in EDIB command line.
Layer must be set to 2 to get correct connection lines.

b. Draw with three phase connection lines.

–Choose **Conn/Ref Mult.conn.line** from EDIB pull down menu.

–Give start point *x,y*, start pitch *x,y*, end point *x,y*, end pitch *x,y*, no. of Conn. lines *n*
(Note: give x,y coordinates by pointing device)

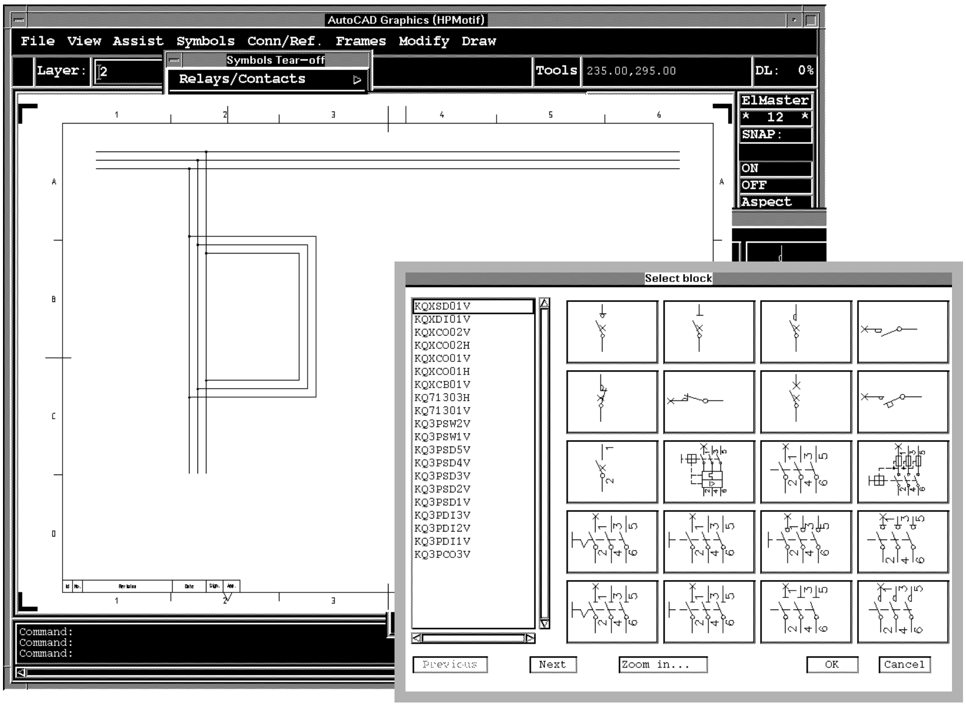
–Repeat function by pressing <**Return**>.



```
End pitch:<5.00,-5.00> :
No. of connection lines: <3>:
Command:
```

Figure 4-12. Drawing multiple connection lines

c. Place symbols on connection lines or append to.
Choose, for example, **Symbols Main circuit Power Circuit switch** from (E)DIB pull down menu and select required symbol from appearing **Select block** icon menu.
Give insertion point by pointing device.



Layer: 2

Tools 235.00,295.00

DL: 0%

Command:
Command:
Command:

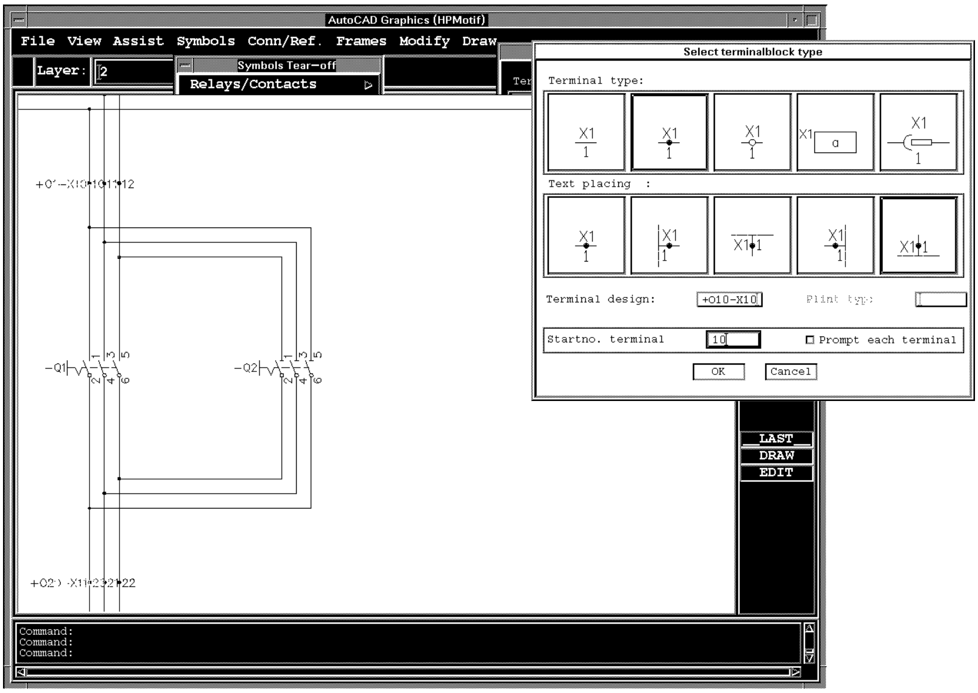
Figure 4-13. Inserting Symbols

d. To place terminal block symbols on connection lines, follow these steps:

– Choose Symbols **Term.blocks Type...** from EDIB pull down menu.

– Within appearing **Select terminalblock type** dialog box select **Terminal type** and **Text Placing**, enter **terminal (bar) designation**, enter **Startno. terminal.**

– Give insertion point of first terminal and distance to next terminal by pointing device, enter number of terminals.



Layer: 2

Terminal design: +O10-X10

Plint typ:

Startno. terminal 10

☐ Prompt each terminal

Command:
Command:
Command:

Figure 4-14. Inserting terminal block symbols

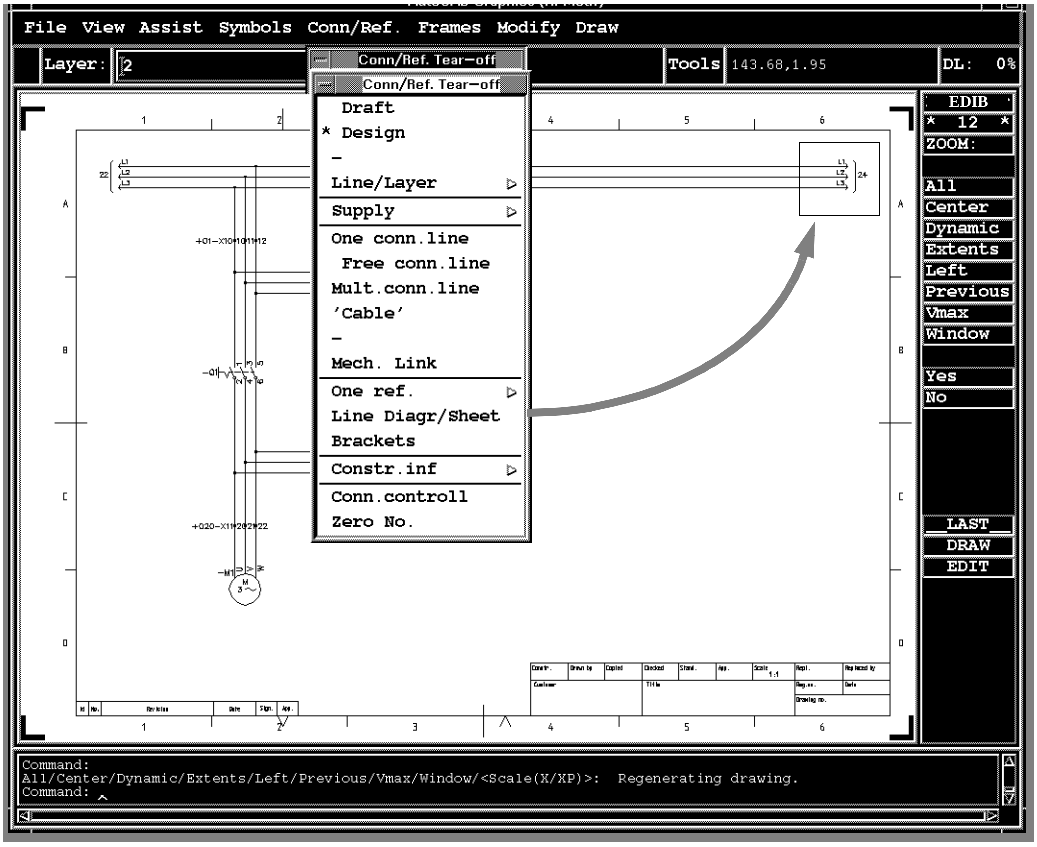
e. To enter sheet references for connection lines, follow these steps:

– Choose **Conn./Ref, Line Diagr/Sheet** from EDIB pull down menu.

– Answer prompt `Enter sheetnumber`: ***`diagram_id/sheetnumber`***.

– Select connection line endpoints with window.

– Answer prompt `Enter the connect. sectionnumber/ID` (for example, L1, L2, L3).



```
Command:
All/Center/Dynamic/Extents/Left/Previous/Vmax/Window/<Scale(X/XP)>:  Regenerating drawing.
Command:
```

Figure 4-15. Entering Diagram/sheet references for connection lines

4.3.1.6 EDIB Layout Operations

NOTE

The EDIB specific functions can only be used in the described way if you have the option Electrical Diagram Builder Layout available. Otherwise see Section 4.3.1.3, Common DIB/EDIB Operations.

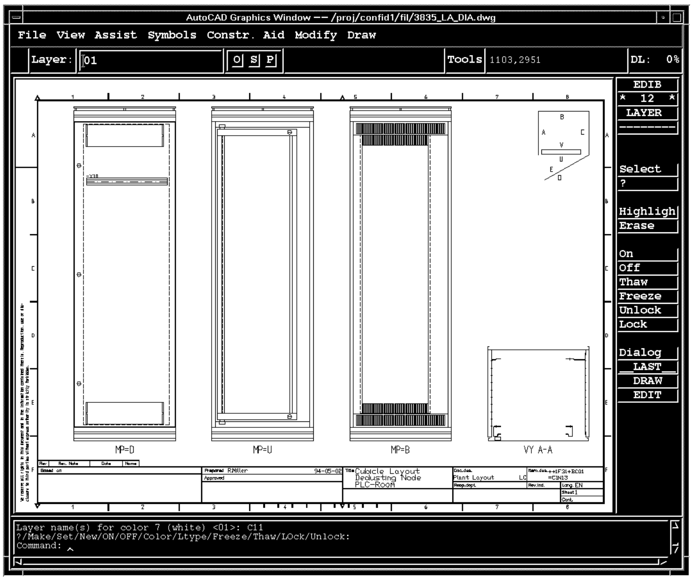
1. Enter/select scale factor
   Layout diagrams are made 1:1 related to reality. If you want, for example, a layout of a cabinet (which may have in reality a height of 2200mm) using an A3L format (which provides on sheet a drawing area height of in maximum 277 mm), you must choose at least a scale factor of 1:10.

2. Insert format (see Section 4.3.1.3, Common DIB/EDIB Operations). Recommended for general layout diagrams is A3L.

3. You can edit the layout diagram with the following steps:

   a. Choose a cabinet symbol from the symbol library via pull down menu item **Symbols Enclosures** and place it.

   b. Choose mounting bar symbol via pull down menu item **Symbols Bars Mounting-bar**, place it, edit item designation and define length.



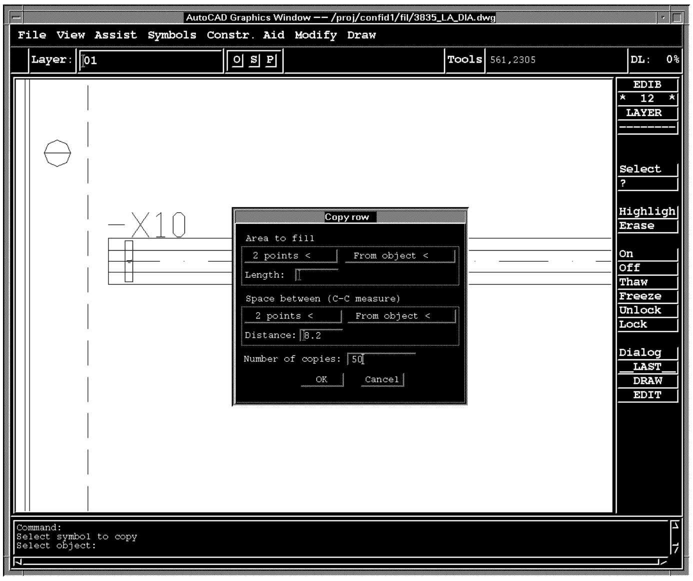
```
Layer name(s) for color 7 (white) <01>: C11
?/Make/Set/New/ON/OFF/Color/Ltype/Freeze/Thaw/LOck/Unlock:
Command:
```

Figure 4-16. EDIB Layout Operations

   c. Choose terminal block symbol via pull down menu item **Symbols Cubicle-mounted Terminal blocks**, place it as first terminal on terminal bar and enter item designation of terminal block.

   a. Copy terminal block by use of pull down menu item **Constr.Aid Copy row.**
   Notes:
   - Make sure symbol related layer group is not locked in AutoCAD.
   - Define **Space between** by choosing **From Object** and selecting terminal symbol.
   - Define 1 as increment number for item designations.

The terminal block symbol is copied and the item designation is incremented automatically.



Layer: 01

561,2305

DL: 0%

Length:

Distance: 8.2

Number of copies: 50

```
Command:
Select symbol to copy
Select object:
```

Figure 4-17. Terminal Block Symbol

4.3.2 Building Symbol Blocks

4.3.2.1 General Aspects

You can build new symbol blocks within a running project "on the fly" within normal diagrams (type CI_DIA,LA_DIA...) and the general Diagram Builder functionality.

For conventions valid within AdvaBuild, see Appendix B and Appendix C of this manual.

4.3.2.2 STB/TEB Operations

Open existing diagram object with view **Diagram.**

4.3.2.3 DIB operations

Setup

Switch on hidden information, use AutoCAD command `layer on`.
For circuit diagram symbol, for example, enter:

- **`layer on $APP`**
- **`layer on NODE_I`**

  If you want to make symbols with a certain module (for example, 2.5 mm), set snap/grid to that module (use AutoCAD command, for example, **`snap 2.5`**)

  If you want to make graphics or attributes on a certain layer, set this layer as default (use AutoCAD command **`layer set`** <*layername*>

Use of Existing Symbols

It is easier to use an existing similar symbol to create a new symbol. Use, for example, DIB menu **Symbols Insert** to select a similar symbol from the library, insert it and explode it. (use AutoCAD command `explode`**).**

Editing Symbols

1. Modify/edit graphic (use AutoCAD commands).

NOTE

Additional allowed grids for graphics are 0.5xM and 0.25xM or 0.125xM or 0.1xM.

2. Modify attributes (use, for example, AutoCAD commands `ddmodify`)
3. Create new attributes (use AutoCAD commands `ddattdef` or `copy,ddmodify`).

Creating Connection Points for Circuit Diagram Symbols

1. Copy/edit connection point definitions (use AutoCad commands `copy, ddatte`).

NOTE

- Connection points are defined by symbolblock NOD.
- For that action snap must be set to 2.5 for correct placement.

2. Copy/edit connection point identifier attributes (use AutoCad commands `copy, ddmodify`).

Creating New Symbol Blocks

1. Select menu items **Symlib, Block create** within Diagram Builder menu bar.
2. Select required symbol library within appearing selector box.
3. Enter Symbol block name within appearing **Block name** dialog box.

   AutoCAD prompts in command line:
   Enter method to create block (?,/Zoom/Zoom?) <Zoom>:
   Enter/Select from menu bar: **Zoom?**

4. Zoom window by pointing device.
5. Enter insertion point by pointing device.

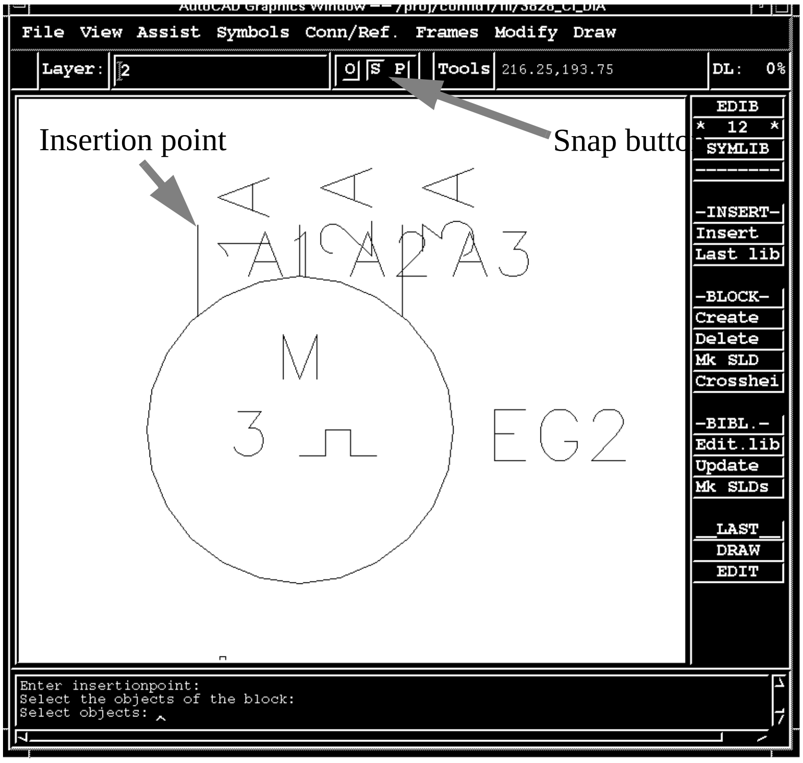
NOTE

Insertion point must be exactly placed on connection point for circuit diagram symbols; recommended is most upper left.

6. Select objects belonging to the symbol block by pointing device.

NOTE

Select attributes in that sequence which is required later within attribute dialog box. Select rest via window. If you have problems to select attributes, switch snap temporarily off by snap-button.



```
Enter insertionpoint:
Select the objects of the block:
Select objects:
```

Figure 4-18. Insertion Point and Snap Button

Creating new Slide Menus

The slides created for the symbol selector box are made from the still exploded symbol. This may not be desired. To correct this, two alternative methods are recommended.

Method A: Update single slide

1. Select corresponding symbol from the library and place it within diagram.
2. Select menu items **Symlib, Block Mk SLD** within Diagram Builder menu bar.
3. Select sublibrary and symbol to be updated.
4. Describe slide window by pointing device and describe insertion point.

Method B: Update all slides within a sublibrary

1. Create a totally empty diagram. To be sure, use AutoCAD command **erase all.**
2. Select menu items **Symlib, Bibl Mk SLDs** within Diagram Builder menu bar.
3. Select sublibrary within **Select symbol library** dialog box.
4. Specify new slides for all blocks within **MasterLine** dialog box.

4.3.3 Building Composed Symbol Blocks

4.3.3.1 General Aspects

You can build new composed symbol blocks within a running project by using normal diagram objects and the general Diagram Builder functionality. Composed symbol blocks can be created just by cutting parts of an existing diagram.

4.3.3.2 DIB operations

Building composed symbols

Use the application specific drawing functions to build up the composed symbol:

1. Make connection lines (for circuit diagrams).
2. Select symbols from the symbol library, place and edit attribute values.
3. Make boundary frames (if required).
4. Select menu items **Symlib, Block create** within Diagram Builder menu bar.
5. Select symbol sublibrary for composed symbol blocks (for example, groups, pgroup) within selector box.

NOTE

> Symbol blocks included in sublibraries for composed symbol blocks are exploded automatically after insert. Therefore, don't mix simple symbol blocks and composed symbol blocks within one sublibrary.

6. Enter composed symbol block name within **Block name** dialog box.
7. AutoCAD prompts in command line:
   Enter method to create block (?/Zoom/Zoom?) <Zoom>:
   Press <**Return**>.
8. Zoom window by pointing device.
9. Enter insertion point by pointing device.

The selected elements are grouped now and stored as one composed block in the symbol library and can be reused by selecting them from a slide menu.

4.4 Operation Instructions

4.4.1 Configuring AutoCAD

See Section 3.5.4, Configuring AutoCAD**.**

4.4.2 Configuring Diagram Builder

See:

- Section 3.4.2, Installing Symbol and Format Libraries in Project
- Section 3.4.3, Adaptation of title block mapping in Project
- Section 3.5.1, Symbol and Format Library handling functions
- Section 3.5.2, Configuring (Electrical) Diagram Builder
- Section 3.5.3, Configuring Plotter Settings.

4.4.3 STB/TEB Operations

4.4.3.1 Inserting Diagram Objects in AdvaBuild

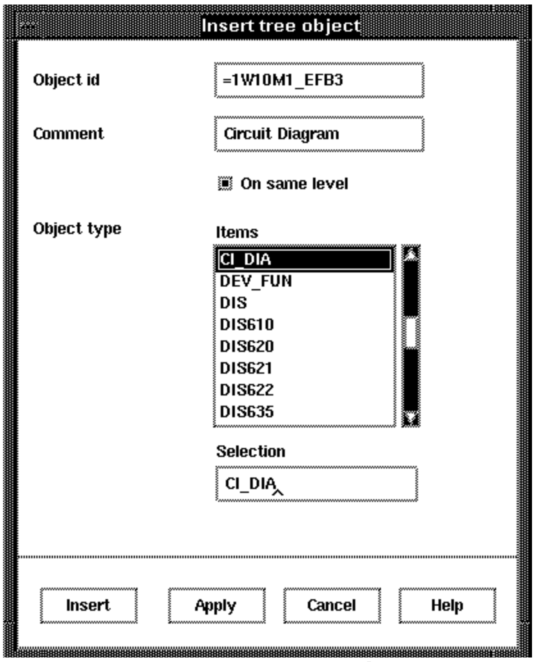
You can insert diagram objects into the AdvaBuild data base via Structure Builder or Template Builder.To insert a diagram object via Structure Builder:

1. Select the desired parent object in the tree representation. Optionally switch to list view (choose **View List** from STB menu).

NOTE

It is recommended to insert diagram objects in list representation.

2. Choose **Edit** from the Structure Builder menu bar.
3. Choose **Insert** from the **Edit** pull-down menu.
4. Select the desired object type from the list displayed in the selection box.
5. Edit **Object id**
6. Click on the **Insert** button to insert the diagram object.



Object id: =1W10M1_EFB3

Comment: Circuit Diagram

☒ On same level

Object type

Selection: CI_DIA

Figure 4-19. Insert Dialog Box

NOTES

- Up to this point, the diagram object includes no graphical content.
- Diagram objects inserted via Template Builder are inserted as list objects.

Naming Conventions:

If, within the object id, certain naming conventions are followed, substrings within the object id can be used to automatically fill up some title block attributes. This functionality is based on the naming conventions described in [I7], [D4].

The rules for the extractable information from the diagram object id and its default configuration for mapping to title block attributes are described in the following figure:

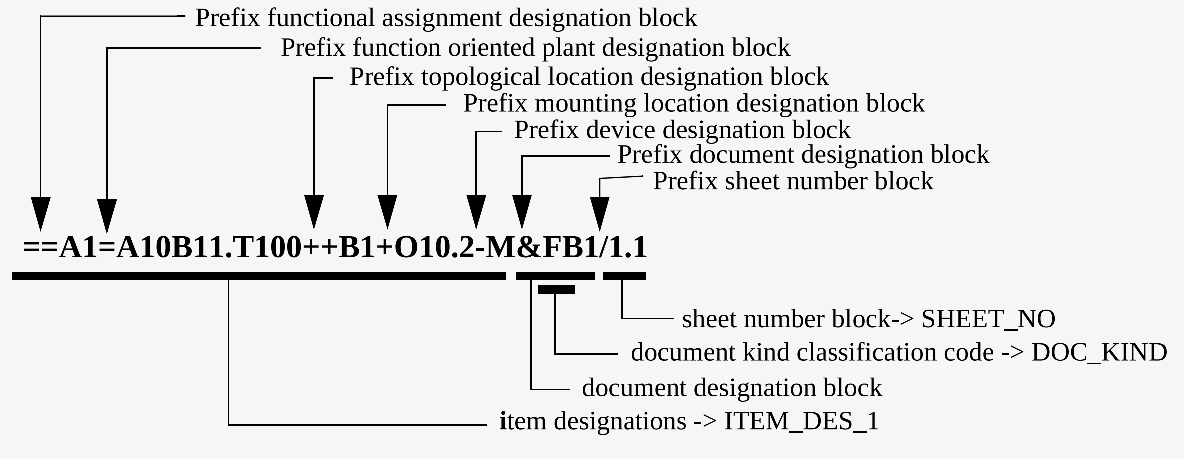

Prefix functional assignment designation block
Prefix function oriented plant designation block
Prefix topological location designation block
Prefix mounting location designation block
Prefix device designation block
Prefix document designation block
Prefix sheet number block
==A1=A10B11.T100++B1+O10.2-M&FB1/1.1
sheet number block-> SHEET_NO
document kind classification code -> DOC_KIND
document designation block
item designations -> ITEM_DES_1

Figure 4-20. Conventions for diagram object ids and mapping to title block

Rules for designation block identification within the object id are:

- Only the valid prefixes ‘==’,’=’,’++’,’+’,’-’,’&’,’/’ define an interpretable designation block.
- Guilty characters within designation blocks are only ‘A’...’.Z’,’0’....’9’,’.’
- Any not guilty character means the end of a designation block.

Since the user is free to disregard these conventions:

- Any designation block may be missing.
- The object id can be built without using any designation block.
- The mapping rules between designation blocks and title block attributes can be customized for diagram object type and project specific within the corresponding configuration file.

4.4.3.2 Editing Diagram Object Attributes

All diagram object types have the identical type and number of attributes and an identical behavior.

Table 4-1. List Of Internal Variables Which Are Preset By Dib

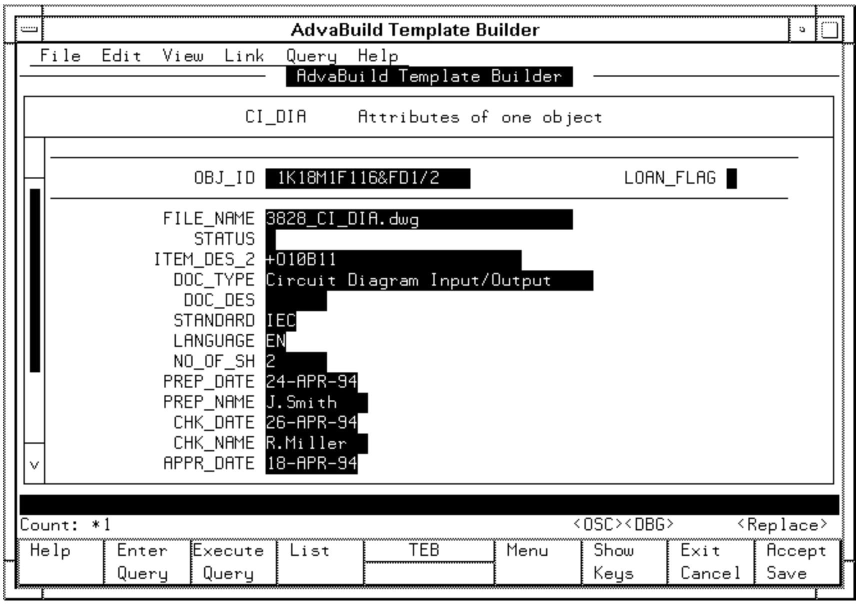
Attribute	DESCRIPTION
Part “File Attributes”	
FILE_NAME	Filename of AutoCAD Drawing File (read-only, set by system)
FILE_FORMAT	Internal File Format (only ‘dwg’ possible)
STATUS	working status (for free use)
Part “Title block 1”	
ITEM_DES_2	Item designation 2 (updated in title block)
DOC_TYPE	Document type description (used as key for document package)
DOC_DES	Document type designation code
STANDARD	Used standard for diagram layout (used for frame/symbol library setup)
LANGUAGE	Language used (used for frame/symbol library setup)
NO_OF_SH	Number of sheets within diagram (for manual setup)
Part “Title block 2”	
PREP_DATE	Prepared date (set to actual date during INSERT/COPY)
PREP_NAME	Prepared name (reset during COPY)
CHK_DATE	Checked date (reset during COPY)
CHK_NAME	Checked name (reset during COPY)
APPR_DATE	Approved date (reset during COPY)
APPR_NAME	Approved name (reset during COPY)

Table 4-1. List Of Internal Variables Which Are Preset By Dib (Continued)

Attribute	DESCRIPTION
Part "Title block 3"	
BASED_ON	(Document is) Based on
REPLACES	(Document) Replaces
REPL_BY,	(Document is) Replaced by
FREE_DIS	for free disposition

Use Template Builder to edit diagram object attributes:

1. Select the desired object in STB.
2. Choose **Object** from the Structure Builder menu bar.
3. Choose **Open** from the **Object** pull-down menu.
4. Select **View Template** from the current selector box to start Template Builder with the template for the selected object type.



```
CI_DIA     Attributes of one object

     OBJ_ID  1K18M1F116&FD1/2          LOAN_FLAG

  FILE_NAME 3828_CI_DIA.dwg
     STATUS
 ITEM_DES_2 +010B11
   DOC_TYPE Circuit Diagram Input/Output
    DOC_DES
   STANDARD IEC
   LANGUAGE EN
   NO_OF_SH 2
  PREP_DATE 24-APR-94
  PREP_NAME J.Smith
   CHK_DATE 26-APR-94
   CHK_NAME R.Miller
  APPR_DATE 18-APR-94

Count: *1                          <OSC><DBG>      <Replace>
```

Figure 4-21. Template Builder in Single Element Mode

NOTES

The attributes STANDARD and LANGUAGE must have the correct values before opening Diagram Builder for the first time on the selected object.

The attributes DOC_TYPE must have the correct values before printing the diagrams within a document package.

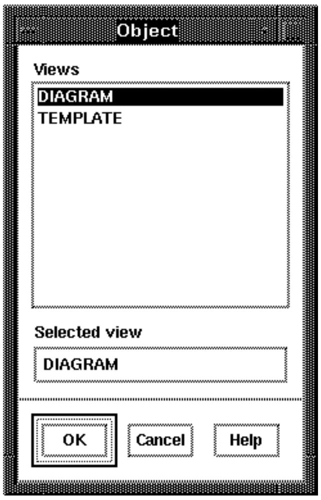
4.4.3.3 Opening Diagram Objects with Diagram View

NOTE

> Before starting the (Electrical) Diagram Builder the first time see Section 3.3, Application Start-up.

To open a diagram object's graphical content, use Diagram view.

1. Select the desired object in the tree or list representation.
2. Choose **Object Open** from the Structure Builder menu bar. Alternatively, double click on diagram object.
3. Choose **View Diagram** from the selector box.



Selected view: DIAGRAM

Figure 4-22. View Selector Box

If the diagram object has been inserted and is opened the first time, there is no format defined within the diagram. For further information see Section 4.4.4, DIB Operations.

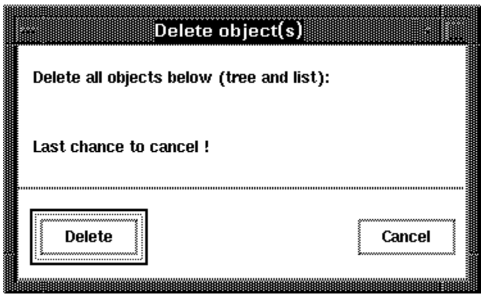
To exit Diagram Builder, use the normal **End** or **Quit** commands.

Although a diagram can be viewed by more than one user at the same time, it can be edited by only one user at a time. If a diagram is already open via Diagram view, a second opening of this object via Diagram view will be in the read-only mode. In this case, you can edit and draw, but all changes will be deleted after you exit AutoCAD.

4.4.3.4 Deleting Diagram Objects

Delete diagram objects using Structure Builder or Template Builder.

1. Select the desired parent object in the tree representation.
2. Choose **Edit** from the Structure Builder menu bar.
3. Choose **Delete** from the **Edit** pull-down menu.



Delete object(s)

Delete all objects below (tree and list):

Last chance to cancel !

Figure 4-23. Delete Object(s) Dialog Box

4. Click on the **Delete** button in the Delete Object(s) dialog box to remove the selected diagram object and the file (including the graphical content) from the data base gand the file system.

4.4.3.5 Copying Diagram Objects with Structure Builder

You can copy diagram objects completely using Structure Builder, which means that:

- All attribute values are copied, and
- Drawing contents are copied.

Diagram objects can be copied individually or as children within a subtree that is copied.

For further information on how to copy single objects or whole subtrees, see *AdvaBuild Basic Functions User's Guide*.

4.4.3.6 Copying Diagram Objects with Template Builder

You can copy diagram objects using Template Builder, but only the attribute values will be copied. The graphical content is not copied. The diagram object's status is the same as after insertion.

4.4.3.7 Renaming Diagram Objects

Renaming means to update the object id or comment attribute of the diagram object. You can rename diagram objects using STB **Edit Rename** menu item.

NOTE

- For diagram objects in tree representation this can be done just by editing the fields in the tree box.
- Renaming diagram objects affects the corresponding attribute values within title block.

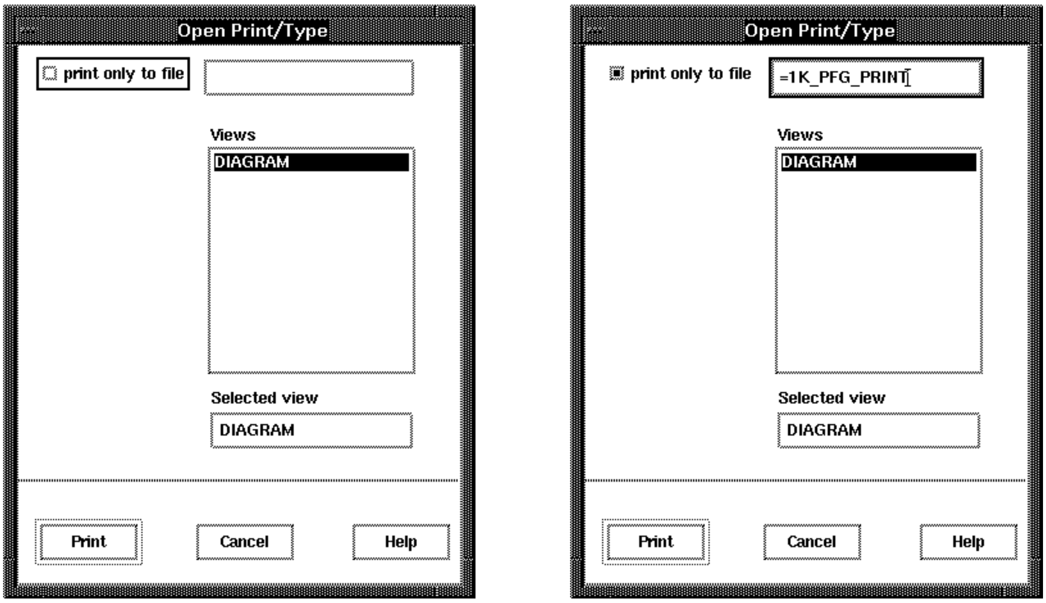
4.4.3.8 Printing Diagrams Directly

NOTE

Before starting the (Electrical) Diagram Builder the first time see Section 3.3, Application Start-up.

To print a diagram directly:

1. Select the desired object in the Structure Builder.
2. Choose **Object Print** from STB menu.
3. Optionally: choose the **print only to file** button and enter the print file name to store the diagram as a postscript file on the user directory.
4. Click on the **Print** button.



Open Print/Type

☐ print only to file

Views: DIAGRAM

Selected view: DIAGRAM

Open Print/Type

☒ print only to file: =1K_PFG_PRINT

Views: DIAGRAM

Selected view: DIAGRAM

Figure 4-24. Print Dialog Box

If you send the diagram directly to the printer, the printer queue corresponding to configuration in the object type specific configuration file is selected. The default configuration queue is **ps**. If you choose **print only to file**, Step 3, a postscript results. The file has the name you entered in the Print dialog box, illustrated above, plus the extension **.ps**.

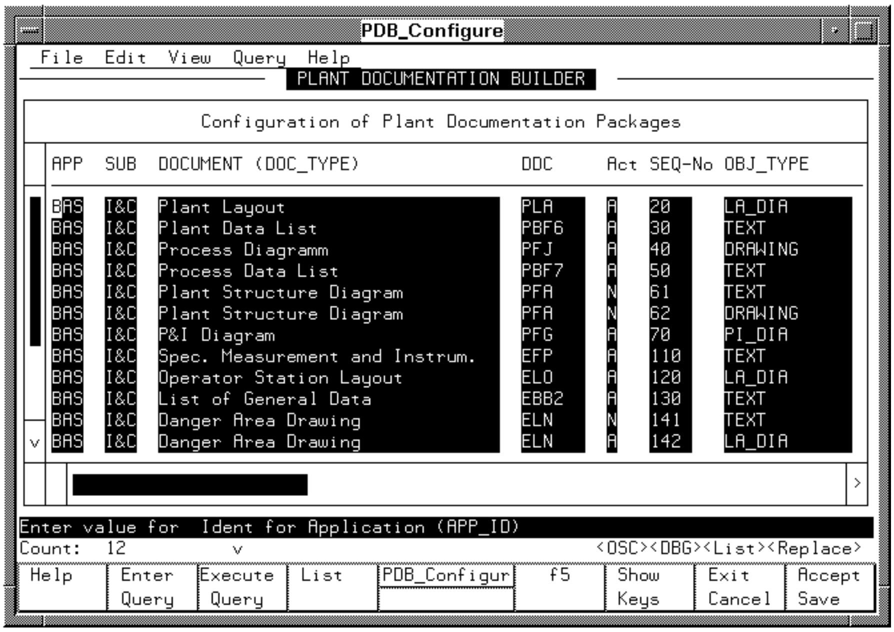
<File Name>.ps

This file is stored in directory:

${C_CAPE}/*<project name>*/usr/*<user name>*

4.4.3.9 Printing Diagrams with Plant Documentation Builder

You can print a number of diagrams automatically within a document package using Plant Documentation Builder.



Configuration of Plant Documentation Packages

APP	SUB	DOCUMENT (DOC_TYPE)	DDC	Act	SEQ-No	OBJ_TYPE
BAS	I&C	Plant Layout	PLA	A	20	LA_DIA
BAS	I&C	Plant Data List	PBF6	A	30	TEXT
BAS	I&C	Process Diagramm	PFJ	A	40	DRAWING
BAS	I&C	Process Data List	PBF7	A	50	TEXT
BAS	I&C	Plant Structure Diagram	PFA	N	61	TEXT
BAS	I&C	Plant Structure Diagram	PFA	N	62	DRAWING
BAS	I&C	P&I Diagram	PFG	A	70	PI_DIA
BAS	I&C	Spec. Measurement and Instrum.	EFP	A	110	TEXT
BAS	I&C	Operator Station Layout	ELO	A	120	LA_DIA
BAS	I&C	List of General Data	EBB2	A	130	TEXT
BAS	I&C	Danger Area Drawing	ELN	N	141	TEXT
BAS	I&C	Danger Area Drawing	ELN	A	142	LA_DIA

Enter value for Ident for Application (APP_ID)

Count: 12 v <OSC><DBG><List><Replace>

Figure 4-25. Plant Documentation Builder

All resulting files are postscript files named by:

<DOC_DES>_<INT_KEY>_<OBJ_TYPE>.ps

where the original drawing file name was:

<INT_KEY>_<OBJ_TYPE>.dwg

NOTE

The value of <DOC_DES> is taken from the Plant Documentation Builder internal table and not from the corresponding attribute of the diagram object.

All resulting files are located in directory:

${C_CAPE}/*<project name>*/doc

and will be sent immediately to printer queue ps.

For details, refer to the *AdvaBuild Basic Functions User's Guide*.

4.4.4 DIB Operations

4.4.4.1 General Operations

Opening a diagram for the first time

If a diagram was inserted by TEB or STB insert, the diagram does not have a format. You must include the format first.

1. Enter/select scale factor (relevant only for LA_DIA, HOOKUP and DRAWING objects).
2. Choose menu items **FORMAT, INSERT.**
3. Choose sublibrary for frames within **Select form setup** selector box (relevant only if several are configured in parallel).
4. Choose sheet format from **Select form** selector box.

4.4.4.2 Editing Drawings

See reference manual *Diagram Builder Functions.*

4.4.4.3 Editing P&I Diagrams

See reference manual *Diagram Builder Functions.*

4.4.4.4 Editing Hookup Diagrams

See reference manual *Diagram Builder Functions.*

4.4.4.5 Editing Layout Diagrams

See reference manual *Diagram Builder Functions.* If the Electrical Diagram Builder option is installed, also see reference manual *Electrical Diagram Builder Functions.*

4.4.4.6 Editing Circuit Diagrams

See reference manual *Diagram Builder Functions.* If the Electrical Diagram Builder option is installed, also see reference manual *Electrical Diagram Builder Functions.*

4.4.4.7 Handling of Item Designations in Circuit Diagrams

Presentation rules for item designations are different for different standards:

What is considered actually in the Electrical Diagram Builder Circuit is IEC 750 for general circuit diagram symbols.

Item designations according to DIN 40719 T3 can only be used in a restricted manner.

General aspects

Within circuit diagram symbols of the Electrical Diagram Builder, there are two separate attributes for handling item designations:

- Attribute ID, called **Item designation**
- Attribute IDF, called **Funct. Item Des.**

These attributes must be used in a certain way to fulfill the naming conventions according to IEC/DIN. Additionally, different boundary frames can be used for detached writing of item designations.

Handling item designations according to IEC 750

According to IEC 750, the product identifying item designation can be built hierarchically in three different ways:

1. The (product identifying) item designation is built up by combination of '-' designation blocks only. This means using the product orientation method. In parallel, an item designation identifying the location can be used.

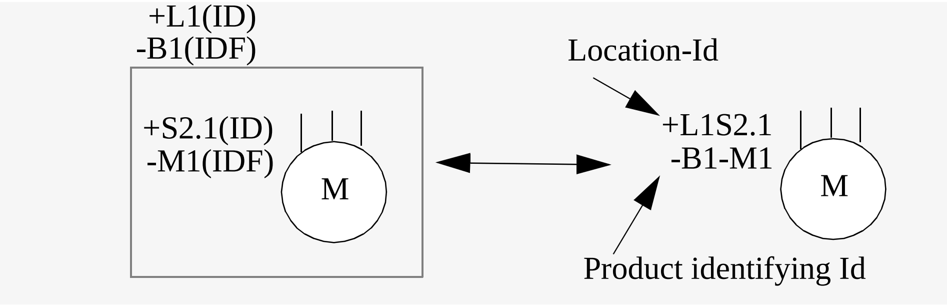


Figure 4-26. IEC Example -A1-M1.

2. If no overordinated '-' item designation but a '=' item designation is available, the item designation is built up by combination of '=' and '-' designation blocks. (This means using the function orientation method.) In parallel an item designation identifying the location can be used.

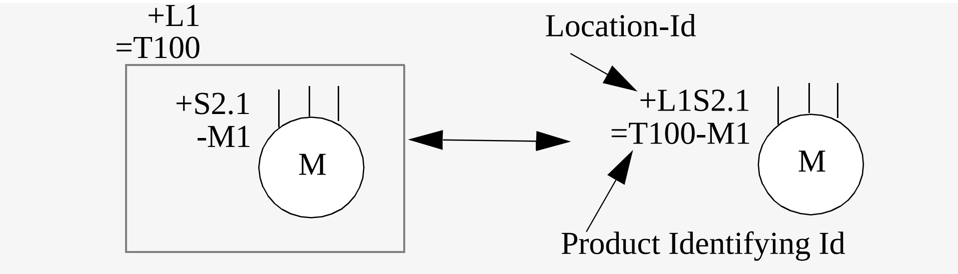


Figure 4-27. IEC Example =T100-M1

3. If no overordinated '=' and '-' item designation but a '+' item designation is available, the item designation is built up by combination of '+' and '-' designation blocks. This means using the location orientation method.

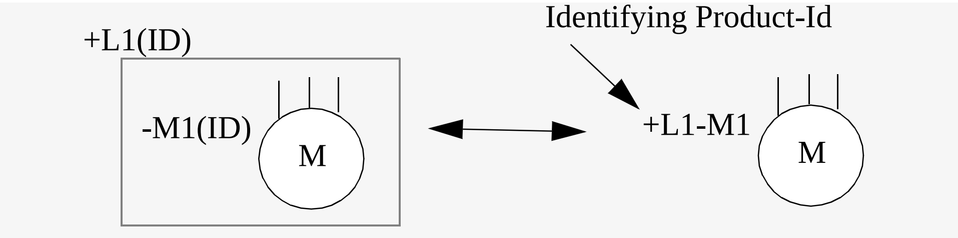


Figure 4-28. IEC Example +L1-M1

4. For absolute item designations, which must not be combined with overordinated item designations, this must be made clear explicitly by use of different line types for the boundary frame.

NOTE

- Linetype "single-dot-dash" indicates that item designations must be combined
- Linetype "double-dot-dash" indicates that item designations must not be combined.

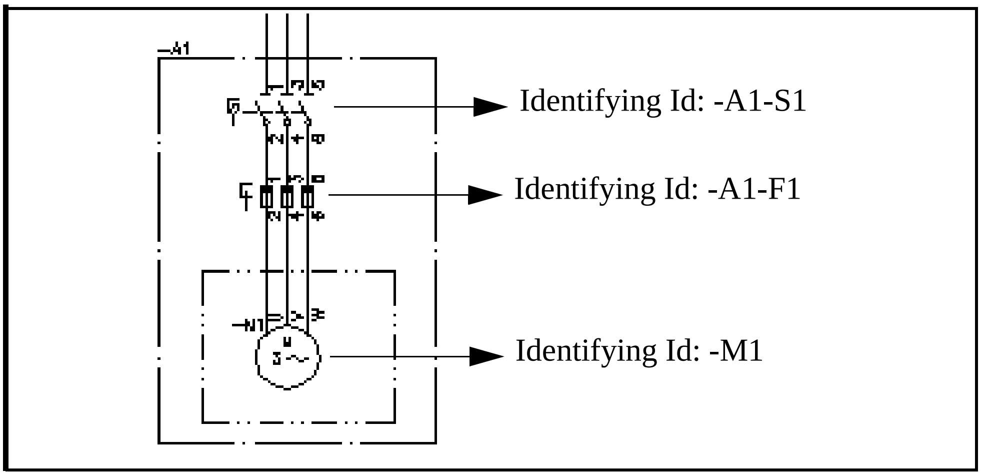


Figure 4-29. Example Identifying Id

Handling item designations according to DIN 40719

The Electrical Diagram Builder Circuit normally does not handle item designations according to DIN. If circuit diagrams will be made according to DIN, you can only apply rules which are common for DIN and IEC.

Therefore, the following restrictions are valid:

- As identifying product item designation, only the following combinations can be used.
  Examples:
  -M1-S2
  =A10-M1
  +O10-M2
- Not allowed with EDIB Circuit, for example, is:
  =A1+O10-M1

Allowed applications of item designations according to DIN are:

1. Building item designations in function oriented way:

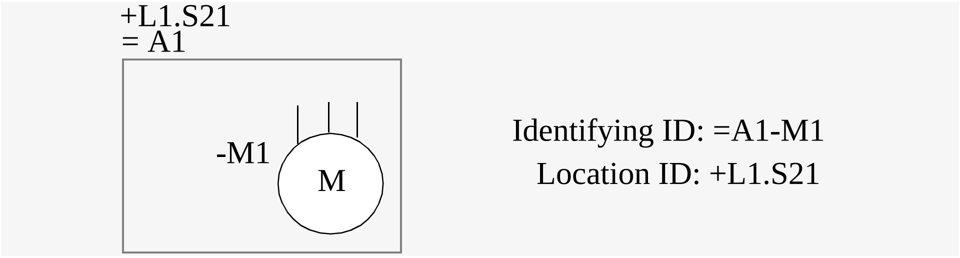


Figure 4-30. DIN Example 1

2. Building item designations in function oriented way:

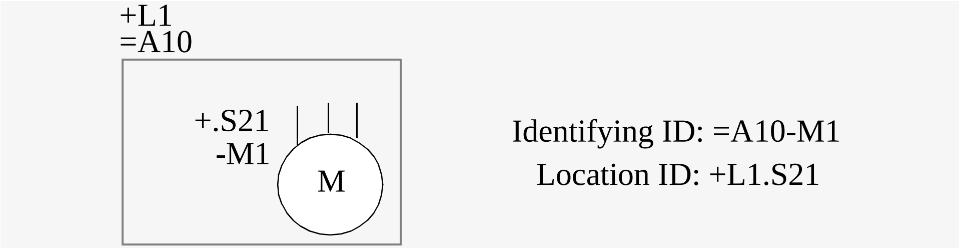


Figure 4-31. DIN Example 2

3. Building item designations in location oriented way:

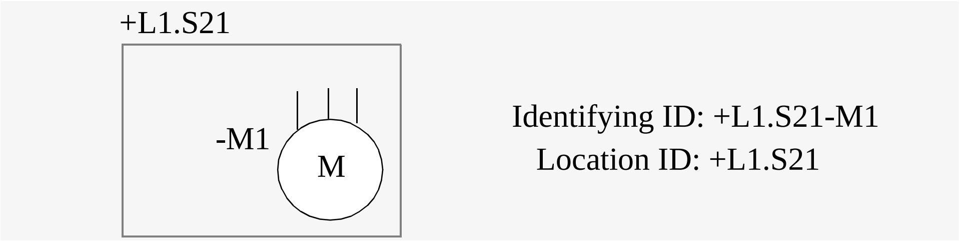


Figure 4-32. DIN Example 3

4. If item designations are not to be combined with item designations on overordinated level (for example, sheet level), boundary frames must be used in the same way as described before for IEC.

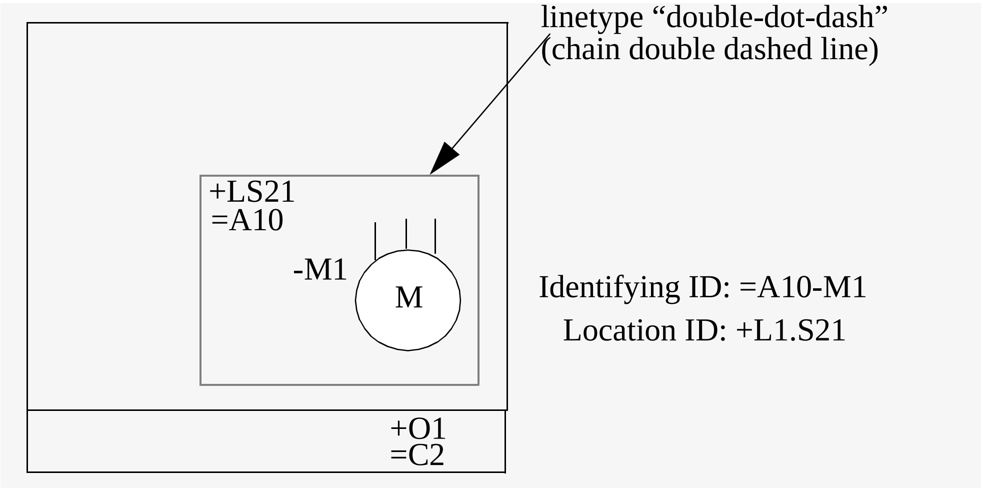


Figure 4-33. DIN Example for absolute item designations

4.5 Operation Menus

In the following figures only the most relevant menus are shown. The dialog boxes, which may come up, are described in the DIB or EDIB reference manual.

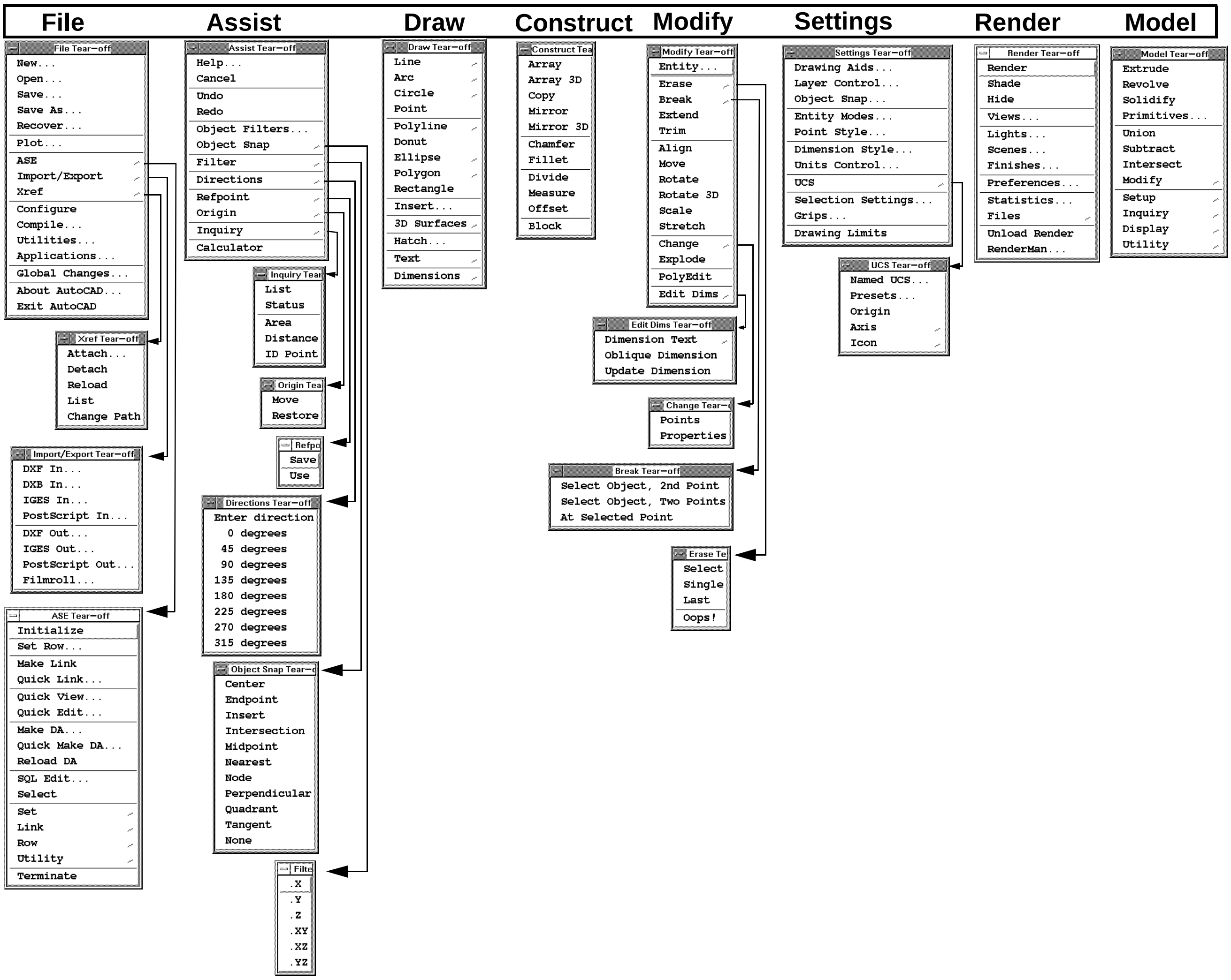


Figure 4-34. (Electrical) Diagram Builder Pull Down Menus

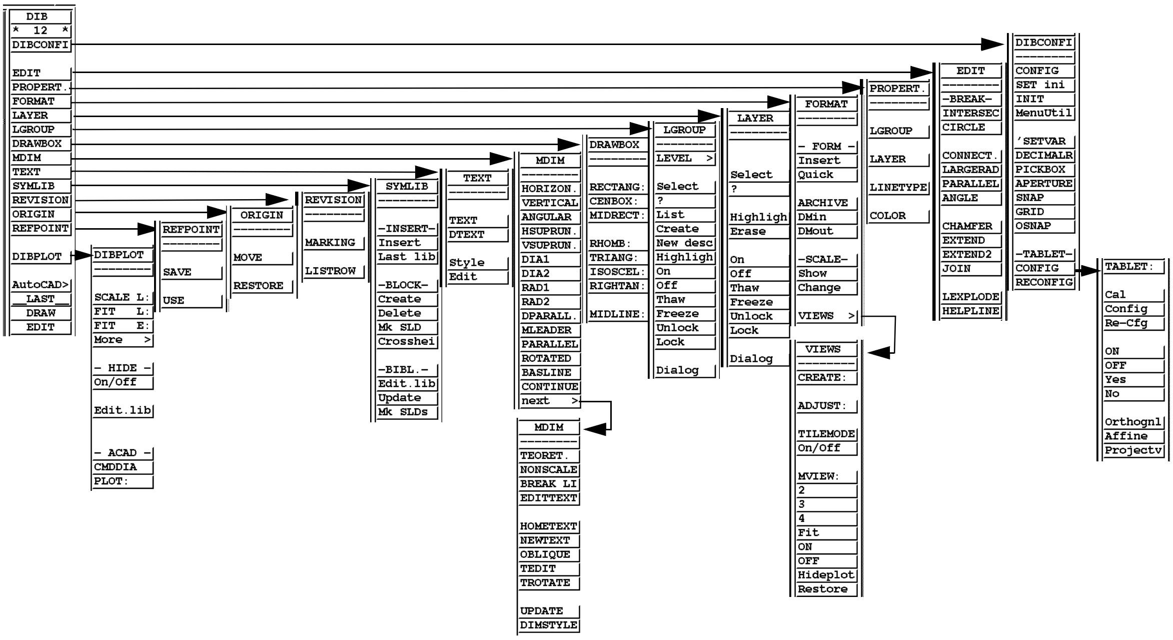


Figure 4-35. Screen menus DIB

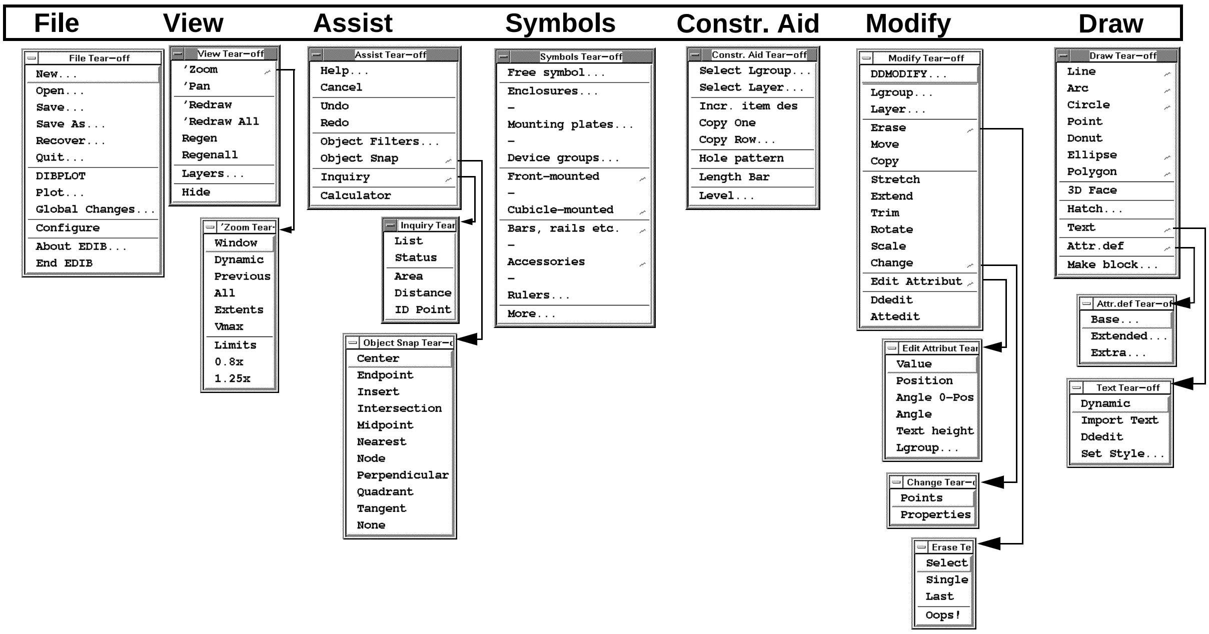


Figure 4-36. Pull down menus EDIB Layout

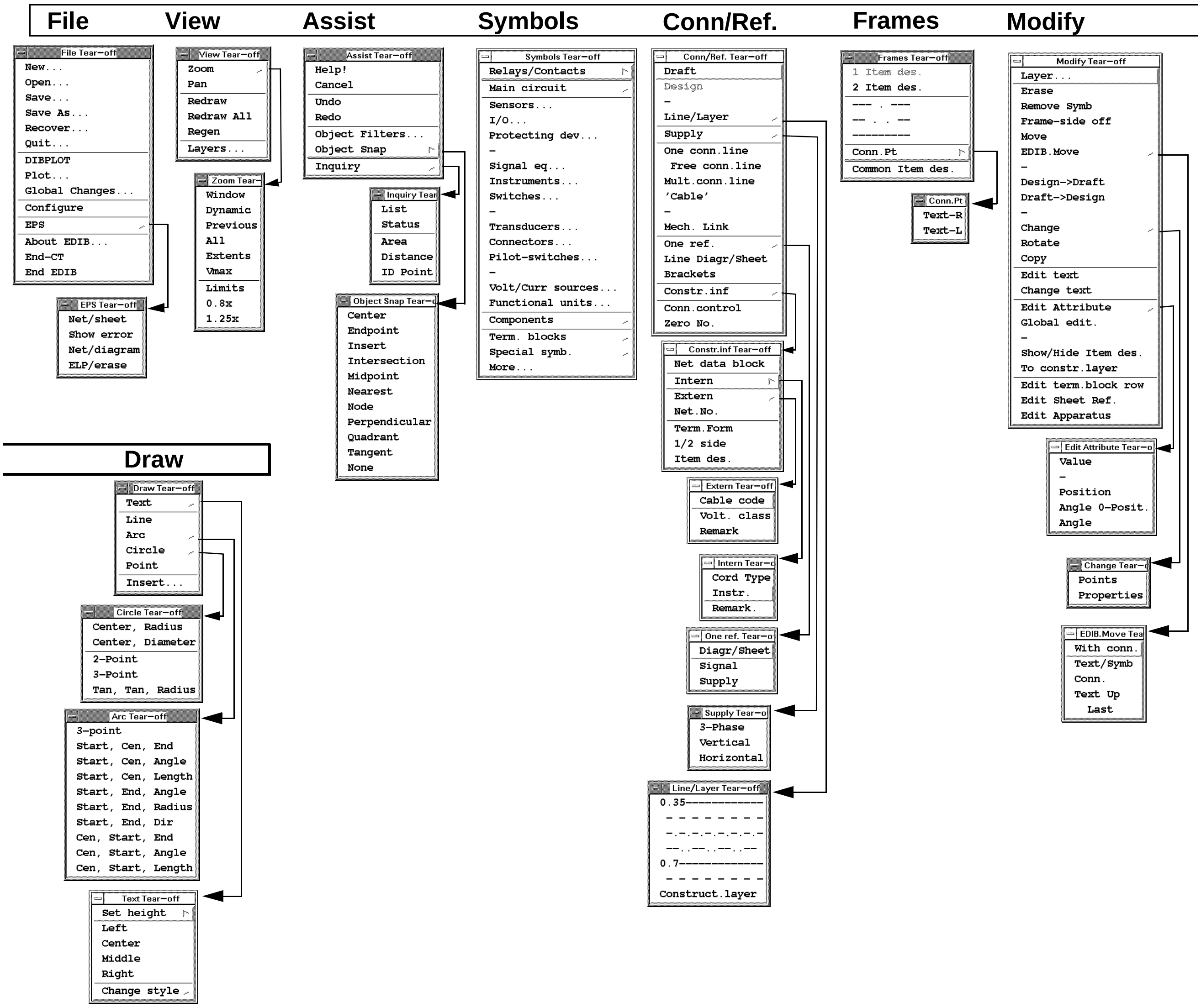


Figure 4-37. Pull down menus EDIB Circuit

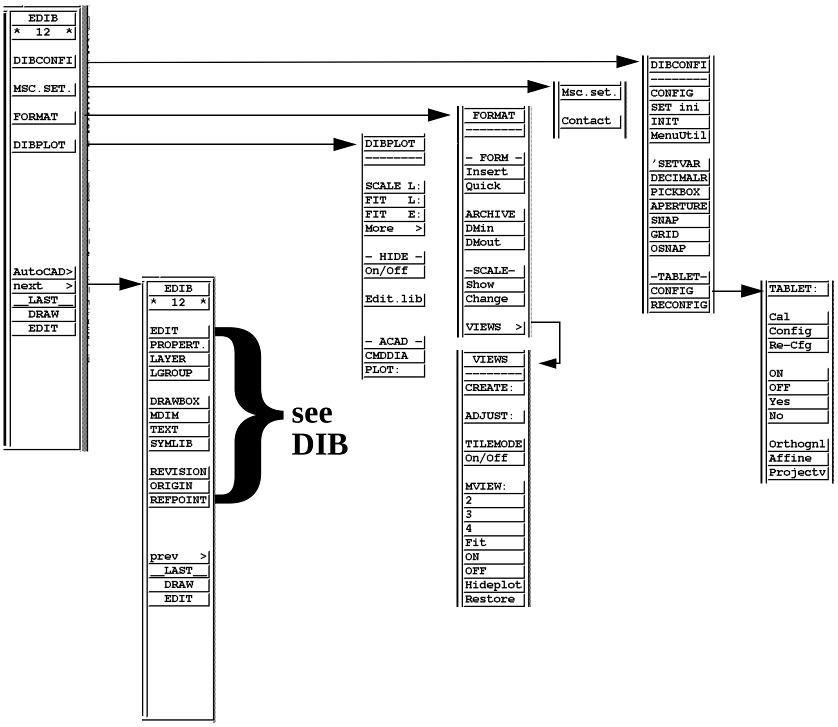


Figure 4-38. Screen menus EDIB Circuit

Chapter 5 Maintenance

5.1 EXPORT/IMPORT of Existing Drawings

You can use AutoCAD to export/import into AdvaBuild. Export/Import is controlled locally within AutoCAD by the Drawing Editor **Exchange** pull-down menu or by inserting a drawing and saving a drawing. AdvaBuild is not directly affected.

You can use the Save function to export a dwg-file or use exchange facilities to get another file format like **dxf**.

CAUTION

Never insert a drawing built with the (Electrical) Diagram Builder directly. This will lead to conflicts with duplicate administration information within the new drawing. First export the drawing contents of the drawing to insert via the AutoCAD command **wblock** and then insert this exported drawing via **insert** (together with **explode**) into the current drawing.

5.1.1 How to Reuse Existing Drawings

The easiest way to reuse a drawing from the same project is to copy the whole object within the AdvaBuild Structure Builder. If you want to reuse a drawing or part of a drawing within another drawing, or you want to reuse an external AutoCAD drawing, use the following procedure:

1. Create a copy of the drawing via AutoCAD command **wblock**.
    a. Start AutoCAD via the Diagram Builder on this object.
    b. Call function **wblock** from the AutoCAD command line. The Create Drawing dialog box appears.
    c. Type in a filename into the File field and click on the **OK** button. The prompt `Block name:` appears at the AutoCAD command line.
    d. Press the **Return** key. The prompt `Insertion base point:` appears at the AutoCAD command line.
    e. Type in **`0,0`**. The prompt `Select objects:` appears at the AutoCAD command line.
    f. Type in **`all`** if you want to reuse the entire drawing or type in **`w`** (window) and select the desired region or click at all entities that should be exported.

       Next, press the **Return** key to leave the object selection. The portion of the drawing that was select is removed from the Graphics Window. You get it back by calling the function **`oops`** from the command line.

   Now the desired objects are written to a file and no unwanted administration information or unwanted objects are included.

2. Insert the created drawing via the AutoCAD function **ddinsert**.
   a. Start the Diagram Builder at the object which should cover the recently created drawing.
   b. Call function ddinsert from the AutoCAD command line. The Insert dialog box appears.
   c. Click on the **Explode** button and then click on the **File** button. The Select Drawing File dialog box appears.
   d. Select the desired file from the File list box and click on the **OK** button. The Select Drawing File dialog box disappears.
   e. Click on the **OK** button. The prompt `Insertion point:` appears at the command line.
   f. Type in **`0,0`** if you want to import an entire drawing or click on the desired position within the Graphics Window if you want to import a drawing part which should be located specifically. The prompt `X scale factor <1> / Corner / XYZ:` appears at the command line.
   g. Press the **Return** key. The prompt `Rotationangle<0>:` appears at the command line.
   h. Press the **Return** key. The particular portion of the drawing appears within the Graphics WIndow.

5.1.2 How to Import a Drawing from a PC

Drawings prepared on a PC will normally be imported into AdvaBuild using a floppy disk drive. A number of functions and a UNIX device are defined for this purpose. For a more detailed description of how to import a drawing from a PC, see the *AdvaBuild Basic Functions User's Guide.*

5.2 Trouble Shooting

5.2.1 (Electrical) Diagram Builder Starts Up With the AutoCAD Menu.

If AutoCAD was not configured as described in the user manual, AutoCAD assumes that the file name case is set to "use lower case". If this happens AutoCAD does not find the current drawing because the drawing name consists of the object type in upper case letters.

Work-around:

Go into the AutoCAD configuration by typing in **config** from the command line. Complete the AutoCAD configuration as described within the (Electrical) Diagram Builder user manual.

5.2.2 AutoCAD Cannot Find/access Existing Files During Insert.

When you insert a symbol in the (Electrical) Diagram Builder AutoCAD may not accept the associated filename or AutoCAD may not find the filename even though the file exists. The problem may originate from the limitation of directory pathname length in AutoCAD. Pathnames for directories containing drawing files must be limited to 65 characters. Within the AdvaBuild directory structure all path names are prepared to be short enough. However you may import symbols etc. from external directory structures which are to long.

Work-around:

If you need a symbol once for insertion, copy it to directory '/tmp/ and then insert from there. If you need symbols more than once, import them into a project specific symbol library.

5.2.3 FILLMODE is set to ON in AutoCAD

When you start the (Electrical) Diagram Builder, the FILLMODE variable is set to ON by default. Sometimes this may cause problems when inserting symbols. Boxes or other properly closed regions may be filled but should not.

Work-around:

Enter **`fill`** in the AutoCAD command line or the AutoCAD Text Window. AutoCAD prompts:

```
ON/OFF <ON>:
```

Enter **`off`** and press <**Return**>. After this enter **regen** and all previously filled regions are now unfilled.

5.2.4 Printing Diagrams from the Plant Documentation Builder or the Structure Builder.

Printing may fail due to some missing configuration within AutoCAD or the (Electrical) Diagram Builder. In some cases the (Electrical) Diagram Builder can not recognize whether an error occurred or whether AutoCAD stops and has a user request. In those cases the icon that comes up on print time is not removed and the print process may never start.

Work-around:

Before you print any diagram verify that used libraries are really installed and that you have completely configured a plotter. After plotter configuration you should make a test printout from the (Electrical) Diagram Builder via the **Print** function in the Structure Builder.

5.2.5 (Electrical) Diagram Builder cannot find liblist.txt files

In AutoCAD error dialog windows a message indicates a certain liblist.txt file cannot be found.

Work-around:

You have assigned some symbol or format libraries in a <object type>.dib file at directory $C_CAPE/<project short name>/msc/dib/cust, but you forgot to install the respective library. See the (Electrical) *Diagram Builder Users Guide* on how to install symbol and format libraries.

5.2.6 Multiple Configuration Points within a Drawing

Once you have inserted a new diagram and opened it via the (Electrical) Diagram Builder, as the drawing content is empty, you may decide to reuse an existing one by just inserting a diagram from directory /proj/<project short name>/fil. Now you will have a problem when starting the (Electrical) Diagram Builder on this object. Any time you do this it will prompt that there are two:

```
2 no. of configurations found in this drawing.
The name of this drawing: 24764_CI_DIA
The following named configurations exist in drawing:
no.: 1 drawing: 24764_CI_DIA
no.: 2 drawing: 24511_CI_DIA
Enter with a number wich configuration that is to be saved:
```

In addition you get the prompt:

```
More than one insertions of the title block found!
Select the title block, or press <Enter> to end function:
```

Work-around:

After the first prompt enter the number that represents the drawing that is mentioned above. After the first prompt select the title block that you have imported with the inserted drawing.

NOTE

Do not use this method to reuse existing drawings. A better way is described below.

1. Open the drawing that you are going to reuse.
2. Write the contents of the drawing that you are going to reuse via **wblock** command or select screen menu item **SYMLIB Create**. If you use menu item **SYMLIB Create** you are asked to which symbol library the drawing contents should be written. If you have specified a special library for such cases you may choose this. If you chose the wblock command, you are free to choose any directory you want. You may use the directory:

   /proj/<project short name>/msc/dib/privat

   to store such blocks for reusing.
3. Now you can reuse those created block any time you want by just inserting them via the **Symbol More...** menu item (in case you used **SYMLIB Create**) or by use of **ddinsert** command (in case you used just **wblock**)

5.2.7 Title Block Update does not Work

You inserted a new format or opened the (Electrical) Diagram Builder and no title block attributes were updated. There is one main reason. The flag **Automatic Updating From Data Base** was switched off. Secondary reasons are:

- The switch was switched off manually once before.
- The current diagram object is from release 2.0/1 and was prepared by the upgrade procedure. The old formats are not prepared for title block update and you should not turn off the update of title block facility.
- You once generated slides for a symbol library from within an (E)DIB session that had this drawing as its current drawing. This procedure erases all information from the current drawing, so the configuration information within the drawing is lost. If you saved this empty drawing the missing configuration information is restored via a default setting. However this does not enable the update of title block information.

Work-around:

In all cases you can solve the problem by selecting the **(E)DIBCONFIG CONFIG** menu item. The Basic configuration dialog box is displayed. Click on the **From Data Base** button to turn on the title block update.

5.2.8 Display of Colors when Working in Parallel with DIB and DYB or DOB

If you work with Diagram Builder in parallel with Document Builder or Display Builder you may notice that colors change as you move the pointer from an AutoCAD window to the window of the other Builder. This is because each Builder uses its own color map. The color map is loaded by the window manager when you move the pointer inside the Builder window. This is not a bug and results from a hardware limitation of the workstation which is only able to load one color map at a time. It is recommended that you first start the Display Builder and then Diagram Builder or Document Builder if you are working at the same time with these Builders.

5.2.9 Creating Slides for Symbol Libraries

Whenever you are ready to create the slides for symbol selection be aware that you either start the (Electrical) Diagram Builder on a special diagram object that is reserved for configuration and slide generation purpose or that you do not save changes when exiting AutoCAD. Exiting AutoCAD and saving the changes to the file system will cause problems in the future.

5.2.10 Reserving an AutoCAD Licence

In some cases, it may be necessary to guarantee that you can get an AutoCAD licence (for instance, if you are printing a document package or if you have to print many diagrams from Structure Builder).

If you already called the (Electrical) Diagram Builder via Open Object then you have reserved a licence. Otherwise you can start AutoCAD from an hpterm by simply calling `acad`. It would be better to first enter: **export ACADDISPLAY=NULL**. This calls AutoCAD and runs it in the current hpterm. In this case no extra window manager resources are used.

However the reservation holds only for the X-Terminal where the mentioned AutoCAD or (E)DIB session was started. Furthermore, each (Electrical) Diagram Builder which is started by another user from the same X-Terminal or another X-Terminal (Console) requires an extra licence.

5.3 Backup/Restore

Backup/Restore of entire projects is maintained by Administration.

Be sure not to create a directory within the project directory that grants no write access to the user **cape**. This will cause problems during Restore. New directories should be created as user **cape** or should be granted write access to user (for example, **master**) but also to the group **dba**.

Appendix A Guideline for Building Formats

A.1 General rules

Paper sheet sizes
(see also Table 1 of ISO 5457/DIN 6771-6)

Table A-1. Relevant paper sheet sizes within AS500ES

Size	trimmed dimensions, mm
A0	841 x 1 189
A1	594 x 841
A2	420 x 594
A3	297 x 420
A4	210 x 297
US letter	215,9 x 279,4

General remark concerning lettering and lines:

The recommended measures depend on the selected module (M). According to IEC standards a minimum module M=2.5 mm is allowed. According to DIN a minimum module M=2.0 mm is possible. The numbers within the following tables are related to minimum module M=2.5mm.

Line thickness:
It is recommended that you use a minimum line thickness of 0.1xM=0,25 mm for A4, A3 and A2 size documents, 0,35 mm for A0 and A1 size documents.

Table A-2. Recommended line thickness within Diagram Builder

	A0	A1	A2	A3	A4
Thick lines	1.0	1.0	0.7	0.5/0,7(1)	0.5/0,7(1)
Medium lines	0.5	0.5	0.35	0.35	0.35
Thin lines	0.35	0.35	0.25	0.25	0.25

(1) 0.5 mm concerning Corporate Standard, 0.7 mm concerning DIN 6771-1

Minimum height of lettering related to sheet size:

Table A-3. Table 2 of ISO 5457

Lettering ISO 3098/10	**A0**	**A1**	**A2**	**A3**	**A4**
A(h = 14d)	5	5	3.5	3.5	3.5
B(h = 10d)	3.5	3.5	2.5	2.5	2.5

Table A-4. Recommended heights within AS500ES, lettertype B

	A0	**A1**	**A2**	**A3**	**A4**
Drawing titles	5	5	3.5	3.5	3.5
Draw/Doc numbers	7	7	5	5	5
Other attributes	3.5	3.5	2.5 (=M)	2.5(=M)	2.5(=M)
Label texts	2.5	2.5	1.8	1.8	1.8

Layers and Styles for frames, title blocks and revision blocks

- All lines, texts and attributes should be placed on layer 0. [1)]
- As linetype must be set "by layer"
- Medium/thick lines must be made by polylines
- Recommended style for attributes within title/revision block is "Standard"
- Recommended style for fix label text within title/revision block is "Label".

---

1. In this case during insert of the format, lines, texts and attributes are mapped to the actual layer.

A.2 Standard Title Block Attributes

The insertion point of the title block is the bottom right hand corner.

Defaults for attributes: Style: Standard, Layer: 0, Height: see tab.

Table A-5. Definitions for title block attributes

TAG	**Prompt Text**	**Exceptions**
DOC_NO:	Document number	
SHEET	Sheet number	
NO_OF_SH	Number of sheets	only relevant for DIN
CONT	Continuation sheet	only relevant for IEC
TITLE_1,2,3,4	Title 1,2,3,4	
ITEM_DES_1,2	Item Des. 1,2	
DOC_NAME_1	Doc.type	
DOC_NAME_2	Doc.designation	
DOC_KIND	Doc.kind code	
LANGUAGE	Language	
PREP_DATE	Prepared (date)	
PREP_NAME	Prepared (name)	
CHK_DATE	Checked (date)	if possible in title block
CHK_NAME	Checked (name)	
APPR_DATE	Approved (date)	if possible in title block
APPR_NAME	Approved (name)	
REV_IDX	Revision Index	
BASED_ON	Based on	
REPLACES	Replaces	
REPL_BY	Replaced by	
PRO_NAME_1 .. 3	Project name (1),(2),(3)	
OWNER	Legal owner	
RESP_DEPT	Resp.Department	
FREE_DIS	For free disposition	
CUST_DIS	For customer Disposition	

NOTE

The sequence of attributes within the table corresponds to the sequence within the title block symbol.

A.3 Optional Title Block Attributes

The following additional title block attributes can be used not included in IEC/DIN title blocks corresponding to corporate standard:

Table A-6. Optional title block attributes available in AS500ES

TAG	Prompt Text	Comes from
CUSTOMER	Customer	PROJECT:CUSTOMER
FACTORY	Factory	PROJECT:FACTORY
PRO_NO	Project number	PROJECT:PRO_NO
STANDARD	Standard	diagram:STANDARD
FILE	File name	diagram:FILE_NAME
FORMAT	Layout format	within drawing
SCALE	Scale factor	within drawing
LM_DATE	Last modified date	diagram:MOD_DATE (obj_instance)
PR_DATE	Print date	system clock during printout

A.4 Revision Block Attributes

Revision blocks can be duplicated manually to handle several revisions within a diagram.
A consecutive numbering can be done by attribute REV_NO.

Defaults for attributes: Style: Standard, Layer: 0, Height: see tab.

Table A-7. Definition of revision block attributes in AS500ES

TAG	Prompt Text	Exceptions
REV_NO	Rev.Number	set invisible (for future internal use)
REV_IDX	Rev.Index	
REV_DATE	Rev.Date	
REV_NAME	Rev.Name	
REV_NOTE	Rev.Note	

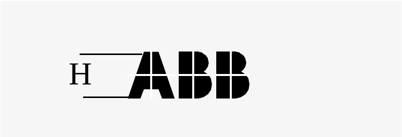
NOTE

The sequence of attributes within the table corresponds to the sequence within the revision block symbol.

A.5 Logo

Rules for ABB Logo

- H = 6 mm ”extra bold” for A4, A3 and A2
  H = 8 mm ”extra bold” for A1, A0
  Design according to Corporate Standard 9ADA 127(B8).



A.6 Grid reference system

- The frame symbol must have folder marks.
- The areas within sheet are identified by a grid reference system.
- The grid shall not be less than 25 mm and not more than 75 mm (IEC 1082-1).

NOTE

Corporate Standard recommends 60-75 mm.

- In X-direction numerical values (1,2,.....) are used for the grid reference system.
- In Y-direction alphabetical values (A,B,....) are used.

Table A-8. Recommended values for grid reference system within Advabuild Diagram Builder

sheet size	**A0L**	**A1L**	**A2L**	**A3L**	**A4L**	**A4P**
X-direction (1,2..)	1 - 24	1 - 18	1 - 12	1 - 8	1 - 6	1 - 4
Y-direction (A,B,..)	A - R	A - L	A - H	A - F	A - D	A - F

A.7 Sheet layout

General rules/restrictions for diagram/drawing formats

- Related to restrictions of printers, the minimum not printable border area is 5 mm.
- To get an original drawing, a layout format must be printable on the corresponding paper size without any scaling.
- The borders have a minimum width of 10 mm for all sizes.

NOTE

For the grid reference system remain therefore 5 mm.

- A filing margin for taking perforation is provided.
  This margin has a minimum width of 17 mm corresponding to IEC.
  This margin has a minimum width of 18 mm corresponding to DIN.
  Within Diagram Builder 20 mm is recommended.

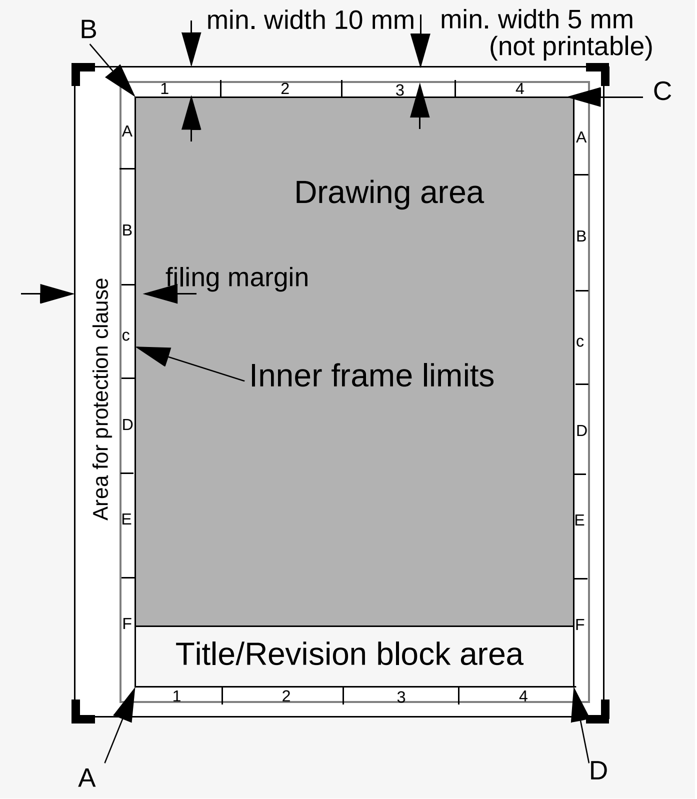


Figure A-1. Sheet layout A4P,Q4P,A3P corresponding DIN/IEC

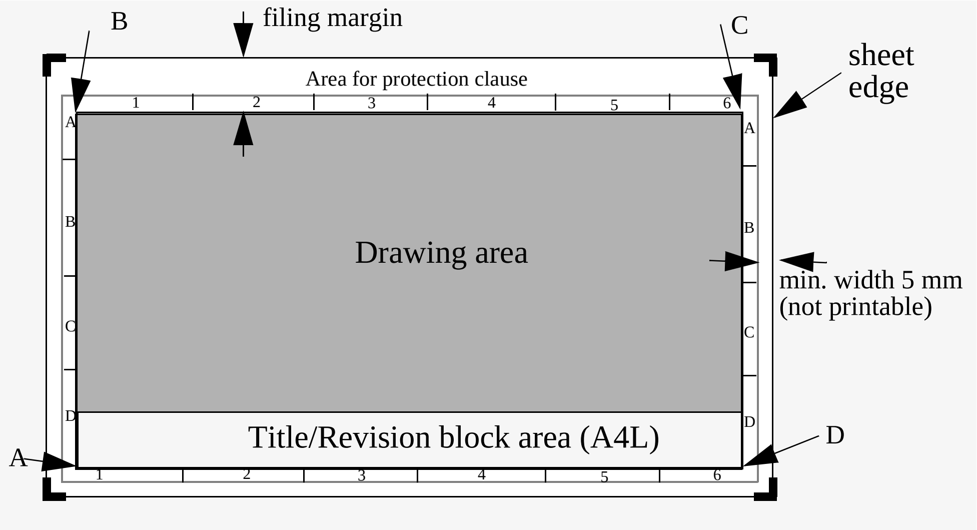

B
filing margin
C
sheet edge
Area for protection clause
1
2
3
4
5
6
A
B
C
D
Drawing area
min. width 5 mm (not printable)
Title/Revision block area (A4L)
A
D

Figure A-2. Sheet layout A4L corresponding to DIN/IEC

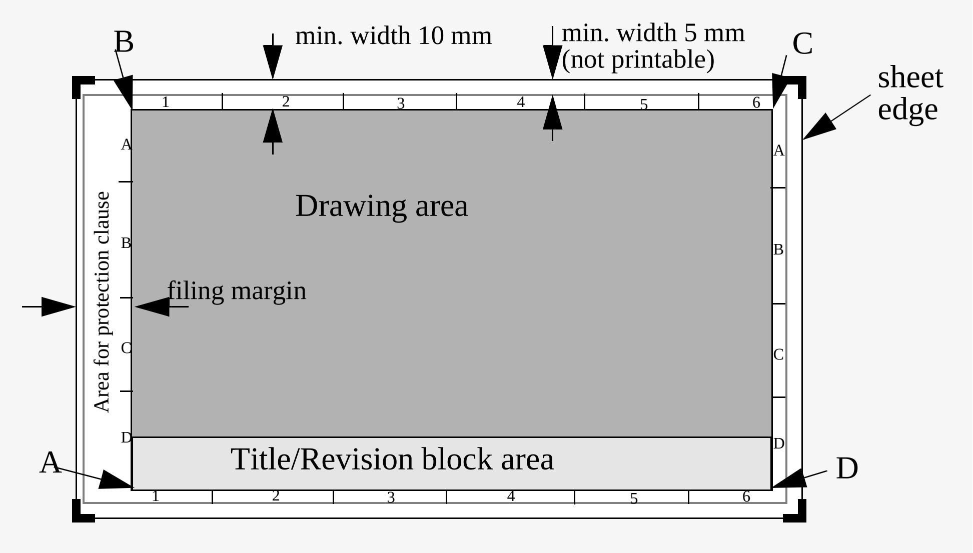

B
min. width 10 mm
min. width 5 mm (not printable)
C
sheet edge
1
2
3
4
5
6
A
B
C
D
Drawing area
Area for protection clause
filing margin
Title/Revision block area
A
D

Figure A-3. Sheet layout A3L corresponding to IEC

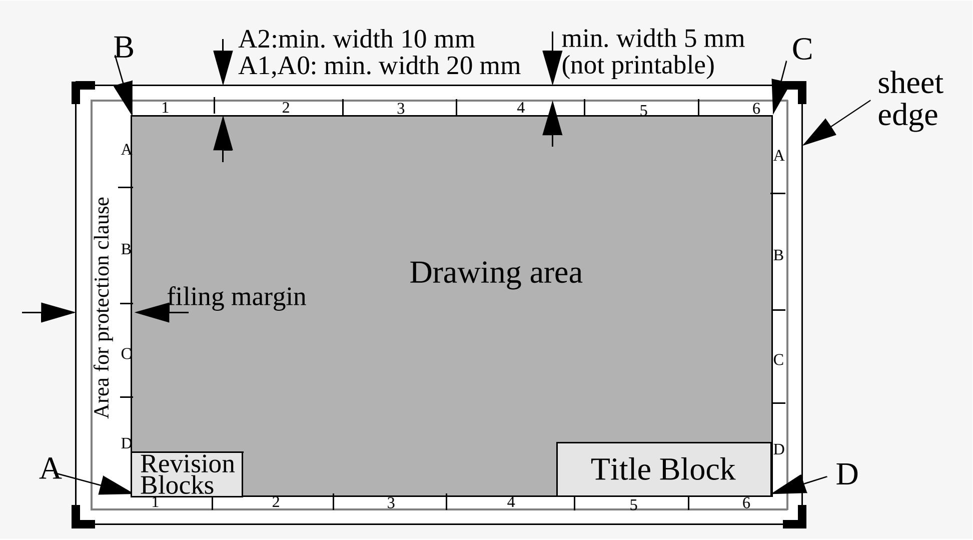


Figure A-4. Sheet layout A2,A1,A0 corresponding to DIN/IEC

Sheet layout A3L corresponding to DIN

For A3L diagrams within DIN 67710-5 an exception is specified related to the other parts of DIN 67710.

The title block width is defined in DIN 67710-5 as 410 mm. But to consider the 5 mm not printable area on each side, it's reduced in AS500ES to 400 mm.

NOTE

DIN 67710 is not consistent in itself. DIN 67710-6 requires a free space of 5mm between sheet edge and grid reference system. In DIN 67710-5 this is not considered. But for practical reasons the free space of 5 mm is needed. The pre-standard DIN V 40 719 T 1000 shows the correct relations (margin 10 mm), but says nothing about the difference to DIN 67710-5 it is based on.

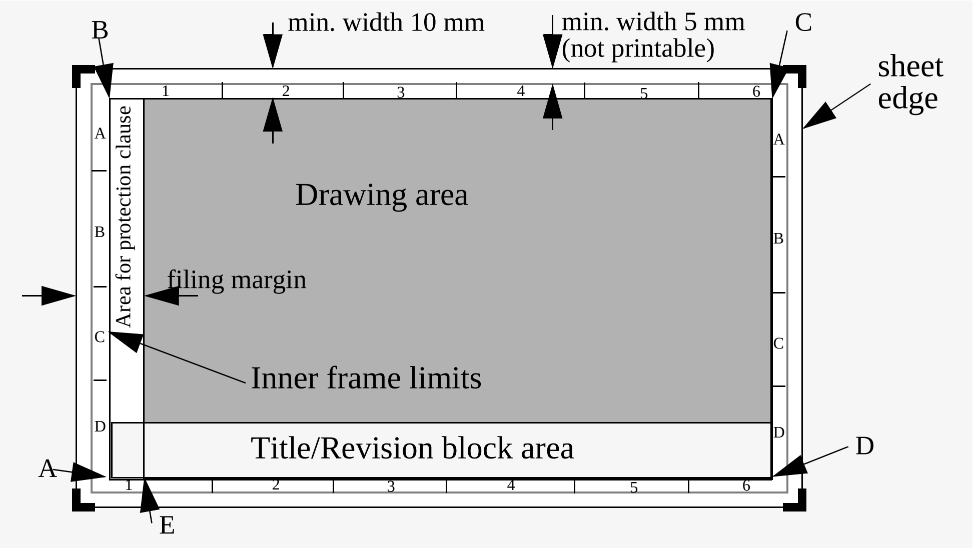


Figure A-5. Sheet layout A3L corresponding to DIN

Table A-9. Recommended coordinates for sheet layout in AS500ES

	A (x,y)	**C(x,y)**
A0L	25,20	1169,821
A1L	25,20	821,574
A2L	20,10	584,410
A3L IEC	20,10	410,287
A3L DIN	10,10	410,287
A3P	20,10	287,410
A4L	10,10	287,190
A4P	20,10	200,287
Q4P	20,10	200,269

NOTE

The common area of US-letter size and A4 size is 210 x 279,4 mm and is called Q4.

A.8 Title/Revision Block Layout

The base point of the title block is bottom right. The insertion point of the title block corresponds to the bottom right point of the drawing area (point D).

If the revision block is handled separately from the title block, the base point of the revision block is bottom left. The insertion point of the revision block depends on layout format and revision block number.

DIN 6771-T1,T5
The distance between lines within title/revision blocks is fixed at 4.23 mm (= 1/6 inch) for A2, A3 and A4. For A0 and A1, 5.6 mm is recommended.

NOTE

> The numbers are based on line distances for line printers. This may not be relevant for modern graphic printers.

ISO WD 7200:1993 (draft)
The distance between lines within title/revision blocks is fixed at 4.5 mm for A2, A3 and A4.

The layout for title/revision blocks for

- A3L corresponding to DIN
- A4L, A4P, A3L, A2L, A1L, A0L corresponding to IEC.

is described in Corporate Standard 9ADA 354.

For A3P corresponding to IEC it is recommended that you use the A4L title block from Corporate Standard 9ADA 354 but reduced to 267 mm.

Alternatively for A2L, A1L and A0L corresponding to IEC the A4P title block can also be used (with adaptations of line distance and letter height).

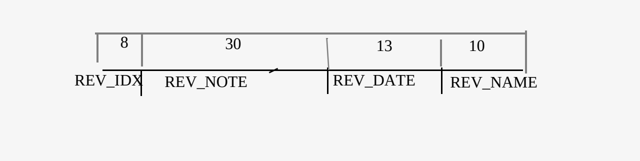
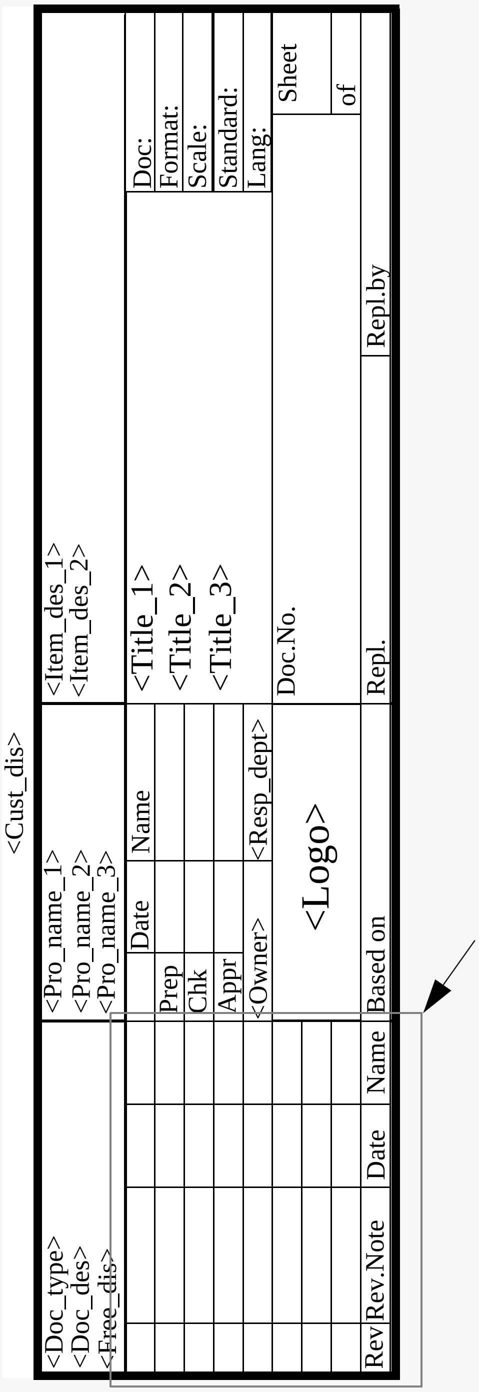


Figure A-6. Example for the layout of a revision block for DIN A3

One or several revision blocks should be inserted already in the completed frame/form.

For A4L/P, A3P, A2L, A1L and A0L corresponding to DIN, see Figure A-7.



Built up by separate revision blocks

Figure A-7. Title block designed corresponding to DIN for A4, A3P, A2, A1, A0

A.9 Scaling, rotation and offset for printout

Original drawings with a certain layout format must be printed/plotted on the corresponding paper size without any scaling (A4P/L -> A4, A3L/P -> A3,....).

For landscape a rotation of 270 degrees clockwise is assumed.

For printing big layout formats on smaller paper formats within the A-series a downscaling must be done. The not printable border of at least 5 mm should be considered.

The mapping for Q4 to A4 respectively US letter paper size is shown in Figure A-8.

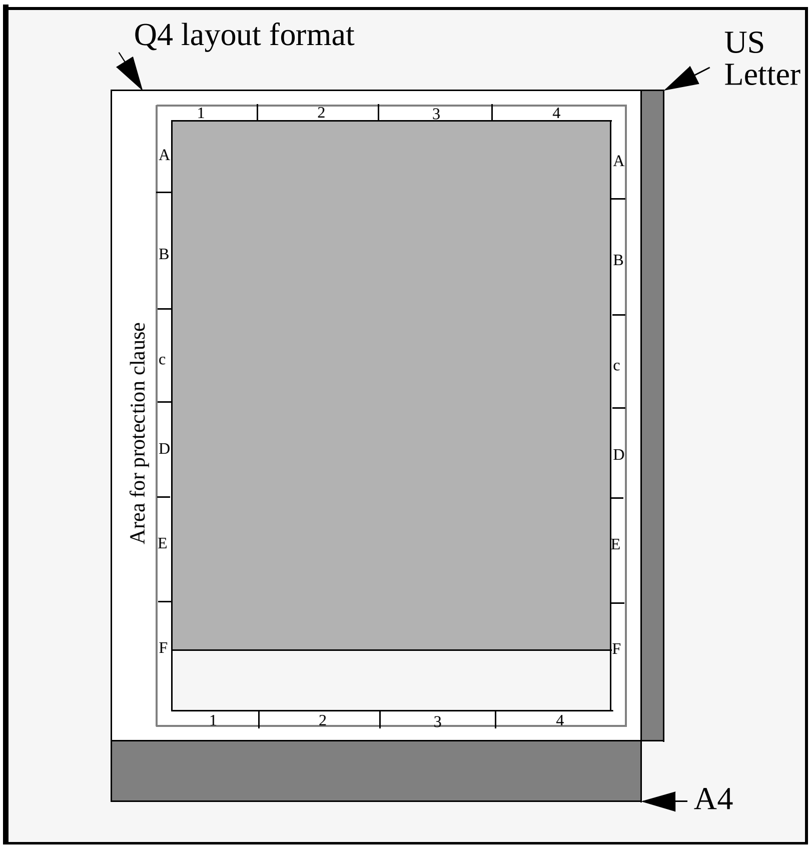


Figure A-8. Mapping Q4 layout format to A4/US Letter paper format

For printing A-layout formats on US letter paper an additional downscaling by factor 0.941 is done.

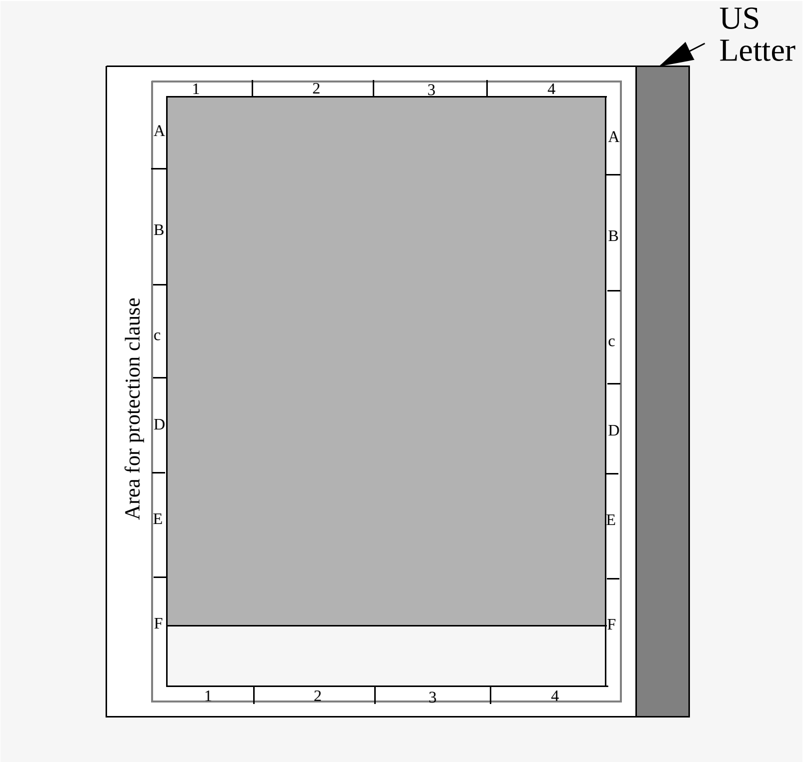


Figure A-9. Printing A-layout formats on US letter paper

A.10 Naming Conventions

The single symbols related to frames/formats should follow these naming conventions:

- 1. Letter: "d" for general drawing symbols
- 2. Letter: "f" for frame symbols
  "t" for title block symbols
  "r" for revision block symbols
  "l" for logosd3. Letter:"i" for iec standard
  "d" for din standard
  "c" for customer standard
  or own defined customer code
- 4-6 letter: layout format, for example, a3l, a4p,...
- 7,8 letter: for free use, may be used for language code, application code or numbering

Example:

- "dtia3le-.dwg" means titleblock corresponding to IEC for A3L format, language English.

Appendix B Structuring and Naming Conventions for Symbol Libraries

B.1 Introduction

B.1.1 Summary

This appendix describes:

- Categories into which symbol libraries must be separated
- Which naming conventions must be followed for defining symbols
- How to organize the symbol libraries on file systems given different Diagram Builder applications.

B.1.2 Background

The symbol name and its file name must describe the meaning and use of a symbol as clearly as possible. Certain rules and restrictions for naming symbols are required. This lets you:

- Sort in new symbols in a correct way
- Set up default rules for selecting symbols
- Look for symbols with certain properties.

B.1.3 Dependencies

There are a number of symbols which do not follow the naming conventions described here exactly. These are mainly:

- Circuit diagram symbols with symbol classification code **k** (first character)
- Some circuit diagram application-specific symbols with symbol classification code **a** not named corresponding to IEC standard
- Layout diagram symbols

Those symbols are today used within hard-coded digitizer menus and specific hard-coded application functions. Therefore, you cannot rename these symbols.

This appendix is only valid if:

- The digitizer menus are not used
- The digitizer menus are adapted for new symbols.

B.2 General Naming Rules

B.2.1 General Symbol Name Conventions

Symbol names must follow certain restrictions. The general rules are described in the following figures.

NOTE

All alphabetic characters must be written in practice as small characters.

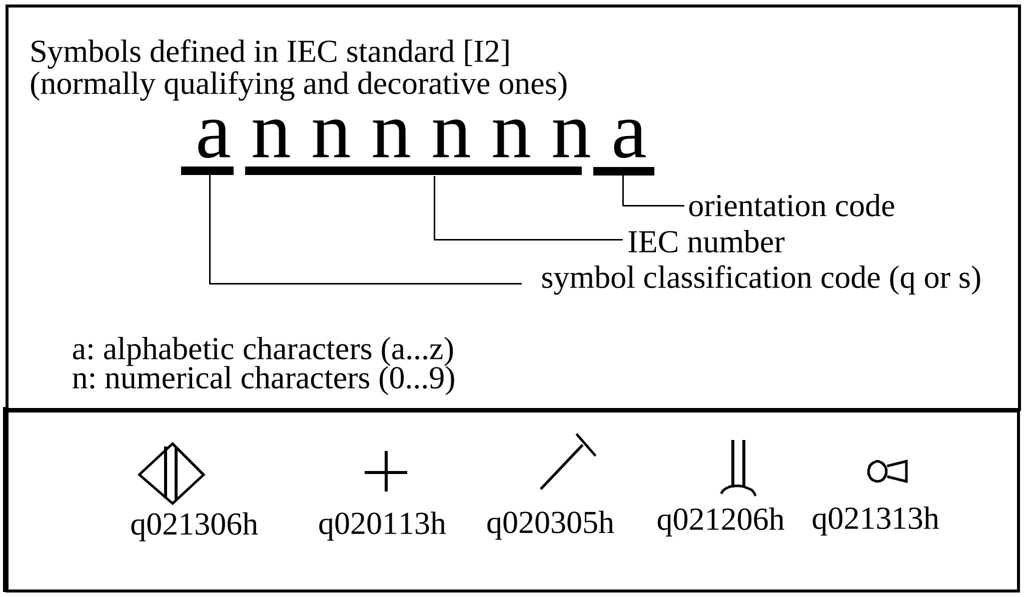

Symbols defined in IEC standard [I2]
(normally qualifying and decorative ones)
a n n n n n n a
orientation code
IEC number
symbol classification code (q or s)
a: alphabetic characters (a...z)
n: numerical characters (0...9)
q021306h
q020113h
q020305h
q021206h
q021313h

Figure B-1. Naming rule I and examples for qualifying symbols corresponding to IEC 617

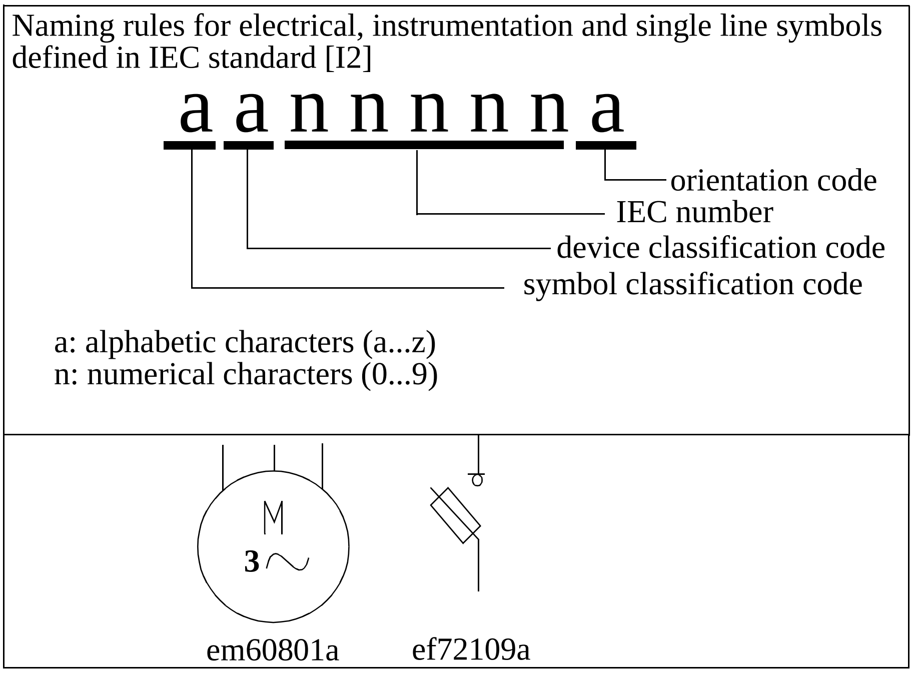

Naming rules for electrical, instrumentation and single line symbols defined in IEC standard [I2]
a a n n n n n a
orientation code
IEC number
device classification code
symbol classification code
a: alphabetic characters (a...z)
n: numerical characters (0...9)
M
3
em60801a
ef72109a

Figure B-2. Naming rule II and examples for electrical symbols corresponding to IEC 617

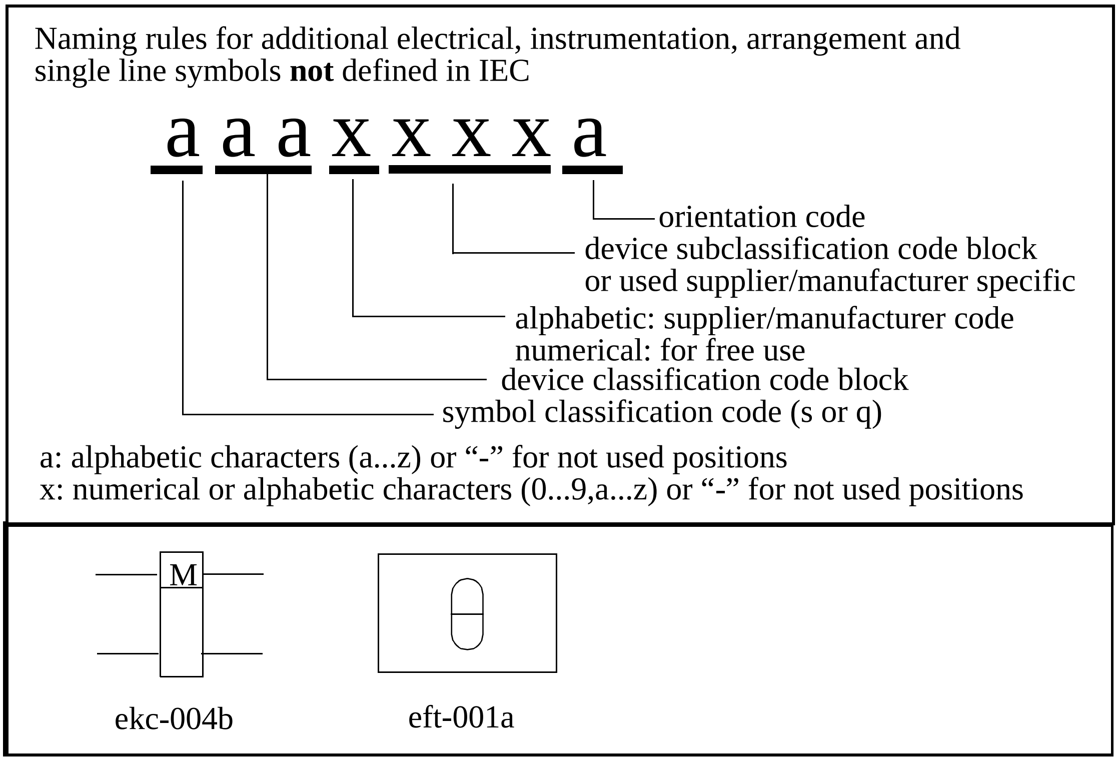

Naming rules for additional electrical, instrumentation, arrangement and single line symbols not defined in IEC
a a a x x x x a
orientation code
device subclassification code block or used supplier/manufacturer specific
alphabetic: supplier/manufacturer code numerical: for free use
device classification code block
symbol classification code (s or q)
a: alphabetic characters (a...z) or “-” for not used positions
x: numerical or alphabetic characters (0...9,a...z) or “-” for not used positions
M
ekc-004b
eft-001a

Figure B-3. Naming rule III and examples for electrical symbols

B.2.2 Symbol Classification Code

Symbols must be designated within a first character corresponding to the following symbol classification codes:

- Electrical symbols for circuit diagrams
  Symbol Classification Code: **E/e**

NOTE

For elder symbols K/k may also be used.

- Instrumentation symbols for circuit diagrams
  Symbol Classification Code: **I/i**
- Cross-reference symbols for circuit diagrams
  Symbol Classification Code: **X/x**
- Mechanical symbols for layout/arrangement drawings/diagrams
  Symbol Classification Code: **L/l**
- Mechanical symbols for hookup diagrams
  Symbol Classification Code: **H/h**
- Block symbols for overview/block diagrams
  Symbol Classification Code: **B/b**
- Single line symbols for overview/single line diagrams
  Symbol Classification Code: **S/s**
- Symbols for P&I diagrams
  Symbol Classification Code: **P/p**
- Decorative qualifying symbols
  Symbol Classification Code: **Q/q**
- Application symbols (specific for all types of diagrams)
  Symbol Classification Code: **A/a**
- Pure graphic symbols for all kinds of drawings
  Symbol Classification Code: **G/g**
- General Diagram Builder symbols (for example, for formats)
  Symbol Classification Code: **D/d**

For future use, the following codes are reserved:

- Mechanical symbols for assembly diagrams
  Symbol Classification Code: **M/m**
- Function block symbols for function charts
  Symbol Classification Code: **F/f.**

B.2.3 Device Classification Code

Within the symbol libraries for electrical, instrumentation, arrangement and single line symbols, the device classification code must correspond to DIN 40719/IEC 750 key codes. Therefore, the second character must correspond to the following conventions:

- A/a: Constructional units, device sets, electronic boards
- B/b: Sensors (non-electrical unit -> electrical unit)
- C/c: Capacitors
- D/d: Digital elements and devices
- E/e: Lamps, heaters, miscellaneous
- F/f: Protective devices, fuses
- G/g: Generators Accumulators
- H/h: Signalling instruments (lamps, horns)
- J/j: For free use
- K/k: Relays
- L/l: Inductors
- M/m: Motors
- N/n: Amplifiers, Controllers
- P/p: Measuring, indicating, testing devices
- Q/q: Power switch devices
- R/r: Resistors
- S/s: Switching devices
- T/t: Transformers
- U/u: Transmitters, Transducers (normally electrical unit -> electrical unit)
- V/v: Semiconductors and electron tubes
- W/w: Cables, other communication transmitting elements
- X/x: Terminals, plugs, sockets
- Y/y: Electrically controlled mechanical devices (actuators)
- Z/z: Filters, limiters.

The device classification code can be expanded by additional alphabetic characters. Conventions are described in Section B.2.6, Refinement of Device Classification Code Block.

B.2.4 Supplier/Manufacturer Code

The supplier/manufacturer code must be used in symbols you define yourself. Use the following conventions:

A/a: ABB

P/p: Phoenix

S/s: Siemens

T/t: Telemecanique

If a symbol is independent/neutral from a supplier/manufacturer, this should be indicated by using '-' at the corresponding position.

B.2.5 Symbol Orientation Code

The last alphabetic character of the symbol name defines the orientation version of the symbol (see [S6]).

- A/a: 0^0, corresponds to definition in standard
- B/b: 90^0 turned anti-clockwise of standard
- C/c: 180^0 turned anti-clockwise of standard
- D/d: 270^0 turned anti-clockwise of standard
- E/e: version A mirrored with regard to a vertical mirroring line
- F/f: 90^0 turned anti-clockwise of version E
- G/g: 180^0 turned anti-clockwise of version E
- H/h: 270^0 turned anti-clockwise of version E.

Example: circuit diagram representation of certain relay coil with 90^0 anti-clockwise rotation sense: ck-c023b.

NOTE

Code E/F/G/H will not be relevant for new layout symbols.

Elder circuit diagram and application symbols with prefix **k** or **a** follow the conventions:

- H/h: horizontal
- V/v: vertical
- U/u: upwards
- D/d: downwards
- L/l: left
- R/r: right.

Elder Layout diagram symbols follow the convention:

- F/f: front symbols.

B.2.6 Refinement of Device Classification Code Block

These definitions are recommendations for symbols used for electrical components and cubicle building.

Only the first character of the codeblock corresponds to the device classification code.
The following characters are recommended for naming rule III (see Figure B-3).

- A: Device sets, constructional units, boards

  aa: electronic boards

  ab: mounting bars

  ac: cubicles, constructional units

  ap: plate symbols

  ar: racks

  ag: device sets/groups.
- B: Sensors

  b: sensors

  be: current/voltage sensor

  bf: flow sensor

  bl: level sensor

  bp: pressure sensor

  bs: frequency, speed sensor

  bx-i: induction sensor

  bx-f: foto sensor.
- C: Capacitors.
- D: electronic boards (DCS/PLC components)

  da: general boards (controller, bus interface, power supply)

  du: I/O boards

  dx: connection units

  dg: composed symbols for device groups (for example, I/O set)

  di: interface/I/O symbols for DCS system

  di-ai: analog input

  di-ao: analog output

  di-di: digital input

  di-do: digital output

  di-g: general signal.

- E: lamps, heaters, miscellaneous

  el: lamp

  eh: heaters.
- F: protective devices

  fe: protective devices current/voltage

  ff: pilot switch flow

  fl: pilot switch level

  fp: pilot switch pressure

  fs-s: pilot switch speed

  fs-a: pilot switch acceleration

  fs-r: pilot switch rotation

  ft: pilot switch temperature.
- G: Generators, Accumulators

  gg: generator

  ga: accumulator.
- H: Signalling instruments

  hl: lamp

  hh: horn.
- K: Relays

  kc: Relay coils

  kx: Relay contacts.
- L: Inductor.
- M: motors

  ma: AC motors

  md: DC motors.
- N: amplifiers, controllers

  na: amplifier

  nc: controller.
- P: measuring, indicating, testing instruments

  pp: indicating, testing panel instrument.
- Q: power circuit devices

  qs: power switches

  qb: breakers.

- R: resistors
- S: switches
  - sp: push-buttons
  - sr: rotary switches
  - sc: timer clock
- T: transformer windings
- U: transducer and transmitters
  - ue: for electrical unit
  - uf: flow transmitter
  - up: pressure transmitter
  - ut: temperature transmitter
- V: semiconductors
  - vt: transistor
  - vy: thyristor
- W: cable symbols
- X: Connective devices
  - xt: terminals
  - xp: plugs
  - xs: sockets
- Y: Actuators
- Z: Filters, limiters
  - zc: C filters
  - zd: Diode limiter
  - zx-rc: R/C filters
  - zx-rd: R/Diode limiter.

B.2.7 Symbol Numbering Concerning IEC 617/DIN

The following list gives only an overview, how symbols are organized within IEC 617

02: Symbol elements, qualifying symbols and other
02-01: Frames, lines, shields
02-02: Current, voltage, potential
02-03: Variability
02-04: Direction of force or motion
02-05: Direction of flow
02-06: Operational Dependence on characteristic Quantity

02-07: Types of material
02-08: Effect or dependence
02-09: Radiation
02-10: Signal waveforms
02-11: Printing, perforation and facsimile
02-12: Mechanical controls
02-13: Operation devices and methods
02-14: Control by non-electrical quantities
02-15: Earth and frame connections, equipotentiality
02-16: Ideal circuit elements
02-17: General application/Misc. symbols
03: Conductors and Connecting Devices
03-01: Conductors
03-02: Terminals and connection of conductors
03-03: Connecting devices (plug, socket)
03-04: Cable fittings
04: Passive Components
04-01: Resistors
04-02: Capacitors
04-03: Inductors
04-04: Symbol elements Ferrit cores and magnetic storage matrices
04-05: Ferrit cores
04-06: Magnetic storage matrices

04-09:
05: Semiconductors and electron tubes
05-01:

05-016:
06: Production and conversion of electrical energy
06-01: Qualifying symbols for winding interconnection
06-02: Internally connected windings
06-03: Elements of machines
06-04: Types of machines
06-05: Direct current machines
06-06: Alternating current commutator machines
06-07: Synchronous machines
06-08: Asynchronous machines
06-09...13:Transformers and reactors

06-14: Power converters
06-15: Primary cells and accumulators
06-16: Power generators
06-17: Heat sources
06-18: Power Generators
07: Switchgear, control gear and protective devices
07-01: Qualifying symbols of Contacts
07-02...06:Contacts
07-07...11:Switches
07-12...13:Complex switches
07-14: Block symbols for motor starters
07-15: Electromechanical relays (relay coils)
07-16...18:Measuring relays and related devices
07-19...20:Proximity and touch sensitive devices/switches
07-21: Fuses
07-23: Gaps and arrestors
07-24: Igniters and flag indicators
08: Measuring instruments, lamps and signalling devices
08-01, 02: Indicating instruments
08-03: Recording instruments
08-04: Integrating instruments
08-05: Counting devices
08-06: Thermocouples
08-07: Telemetric devices
08-08: Electric Clocks
08-09: Misc. measuring elements and instruments
08-10: Signalling instruments (Lamp, horn, siren, buzzer)
09: Telecommunications: Switching and peripheral equipment
09-01...04: Switching systems and equipment
09-05...07: Telephone, telegraph and data equipment
09-08...10: Transducers, recorders and reproducers
10: Telecommunications: Transmission
10-01...02:Telecommunication circuits
10-03...06:Antennas and radio stations
10-07...11:Microwave technologies
10-12...20:Misc. block symbols
10-21...22:Frequency spectrum diagrams
10-23: Fibre optics, Transmission lines
10-24: Fibre optics, Transmission devices

B.3 Application specific Mapping of Symbols to Library (sub)directories

B.3.1 Circuit Diagram Application

Subdirectories, including symbols for circuit diagrams, are all located below directory ${C_PROJ_DIB_LIB}/cis*<standard><language>*/blksld.

NOTE

Elder conventional circuit diagram symbols with prefix k will not follow exactly these conventions.

B.3.1.1 Electrical and Instrumentation Symbols

First character:

e: for conventional circuit diagram symbols (see [I2])
i: for instrumentation circuit diagram symbols (with additional attributes) (see also [S2]).

NOTE

Customer specific symbols with customer code at the 3rd position may not follow these rules in the following positions.

Structuring/naming conventions (2...5 character)to subdirectory

ag*: composed symbols for general device sets/groups-> cgroups

b*: sensors -> sensors

c*: capacitors -> passcomp

da*: general DCS boards (controller, bus interface, power supply)-> units

du*: DCS I/O boards-> units

dx*: DCS connection units-> units

dg*: composed symbols for device sets (I/O set)-> cgroups

di*: interface/I/O symbols for DCS system-> i_o_unit

e*: lamps, heaters-> passcomp

fe*: protective devices current/voltage-> protect

ff*: pilot switch flow-> pilotsw

fl*: pilot switch level-> pilotsw

fp*: pilot switch pressure-> pilotsw

fs-s*: pilot switch speed-> pilotsw

fs-a*: pilot switch acceleration-> pilotsw

fs-r*: pilot switch rotation-> pilotsw

ft*: pilot switch temperature-> pilotsw

g*: generators, accumulators-> source

h*: signalling instruments-> signal

kc*: Relay coils-> coils

kx*: Relay contacts-> contacts

l*: Inductors -> inductor

m*: motors -> motors

n*: amplifiers, controllers (not DCS)-> units

p*: measuring, indicating, testing instruments-> instrum

q*: power circuit devices-> pow_circ

r*: resistors -> passcomp

s*: switches -> switch

t*: transformer windings-> transf

u*: transducer and transmitters-> transduc

v*: semiconductors-> semicond

w*: cable symbols-> conduct

x*: terminals -> connect.

NOTE

For selecting and numbering terminals automatically, a predefined algorithm exists within the circuit diagram application, which is based on coded naming conventions. Do not rename terminal symbols with **k** as first character.

y*: actuators -> actuator
(instrum, if not
available)

z*: filter elements/limiters -> filters.

B.3.1.2 Cross-reference Symbols

First character:

x: for cross-reference symbols

Structuring/naming conventions (2...4 character)to subdirectory

kx*:Relay contacts -> xref.

B.3.1.3 Miscellaneous Symbols

The naming conventions for these symbols are defined now only for the first character.

Structuring/naming conventions (1st character)to subdirectory

s: single line symbols-> single

q: qualifying symbols (first character)-> qualif

Remark: Followed by identification number concerning IEC code

a: application symbols (used within functions, must not be renamed!) -> applik.

B.3.2 Layout Diagram Application (Arrangement)

Subdirectories including symbols for layout diagram are all located below directory ${C_PROJ_LIB_DIB}/las*<standard><language>*/blksld.

NOTE

Elder layout symbols having **f** as last character may not follow exactly these naming conventions.

B.3.2.1 Symbols for Arrangement

First character:
l: for layout symbols used in layout/arrangement drawings

In the following text the appended device classification code (block) is specified and the mapping to file system directories.

Structuring/naming conventions (2...3 character)to subdirectory

ab: mounting bars-> bars

ac: cubicles, constructional units-> cubicle

ag: device sets/groups-> groups

ap: plate symbols-> plates

ar: racks -> cubicle

da,du: boards for DCS/PLC system-> plc

dx: connection units-> plc

el: lamps -> lamp

f: protective devices-> fuse

ga: accumulators-> msc-cub

hl: signal lamps -> lamp

k*: Relays -> auxrelay

n*: amplifiers, controllers (not DCS/PLC)-> msc-cub

p*: general measuring, indicating, testing instruments -> instr

pp: indicating, testing panel instrument -> msc-pan

qs: power switches-> contact

qb: breakers -> breaker

sp: push-buttons-> pushbutt

sr: rotary switches-> rotsw

sc: timer clock -> addition

t*: transformers -> msc_cub

u*: transducer and transmitters (cubicle mounted)-> msc-cub

xt: terminals -> terminal

xp: plugs -> terminal

xs: sockets -> (base).

B.3.2.2 Miscellaneous

First character: 'l'

Structuring/naming conventions (2...3 character)to subdirectory

'nnn'mm: cable ducts-> ducts

-p: misc. symbols for panel assembly-> msc-pan

-c: misc. symbols for cubicle internal assembly-> msc-cub

-a: misc. symbols for additional accessory-> msc-add

-x: symbols for misc. accessory (for example, clock)-> addition

-r: Symbols for current and earth rails-> rails.

B.3.2.3 Recommended Conventions for Not Yet Considered Layout Symbols

First character: 'l'

Structuring/naming conventions (2 character)to subdirectory

b: sensors -> lsensor

c: capacitors -> msc-ele

l: Inductors -> msc-ele

r: resistors -> msc-ele

v: semiconductors-> msc-ele.

B.3.2.4 Symbols for Application Specific Symbols

First character: 'a'

Structuring/naming conventions (2nd character)to subdirectory

m: ruler symbols for layout -> ruler

x: misc. symbols for special applications -> applik

h: hole punches -> holes

l: labels/markers-> label.

B.3.3 Hookup Diagram Application (Assembly)

The emphasis of hookup diagrams is on arrangement/assembly of field instrumentation devices (measuring equipment, actuating equipment).

Subdirectories including symbols for Hookup diagrams are all located below directory ${C_PROJ_LIB_DIB}/hos<*standard*><*language*>/blksld.

B.3.3.1 Mechanical Symbols for Hookup

First character **h** for hookup symbols used in hookup/assembly diagrams.

NOTE

Additionally, layout symbols ('l') and graphical symbols ('g') may be used. For those symbols see naming conventions in the corresponding chapters.

Structuring/naming conventions (2...3 character)to subdirectory

ag: device/component groups-> hgroups

ac: terminal/junction boxes-> hboxes

b*: sensors -> hsensors

m *: motors -> hmotors

w*: cables -> hconduct

x*: terminals -> hterminal

y: actuators -> hactuat

h: pipes -> hpipes

h: valves -> hvalves

h: screws -> hscrews

h: fittings -> hfitting

h: pneumatic components-> hpneum

h: hydraulic components-> hhydraul

h: miscellaneous components-> hmsc-com.

B.3.3.2 Miscellaneous

Structuring/naming conventions (1st character)to subdirectory

q: qualifying symbols-> hqualif.

B.3.4 P&I Diagram Application

The emphasis of P&I diagrams is on describing the process sense/flow and the related process equipment. The control flow is only roughly indicated.

Subdirectories including symbols for P&I diagrams are all located below directory ${C_PROJ_LIB_DIB}/pis<*standard*><*language*>/blksld.

B.3.4.1 P&I Diagram Symbols

First character for P&I diagram symbols is:

- P/p: P&I diagram

The second character can be used according to [S7] or [D4].

The main relevant codes concern ISO:

Structuring/naming conventions (2...3 character)to subdirectory

g: groups/composed symbols/macro-> pgroups

b*,k*,c*: vessels, tanks-> pvessel

w*,d*: heaters, heat exchanger, steam generators, coolers-> pheat

i[1]: pipes, pipe fittings-> ppipe

l[1]: valves, valve fittings-> pvalve

ff: filters -> pfilter

fs: separator, sorting, screening machines-> psep

s*: centrifuges -> psep

t *: drier -> pheat

z *: crushing/grinding machines, mills-> pmachine

r *: agitators/stirrers-> pmisc

e[1]: mixer/kneaders-> pmachine

a *: shaping machines, extruders-> pmachine

p *: liquid pumps-> ppump

v*: compressors, vacuum pumps, blowers, fans-> ppump

h*: lifting, conveying, transport-> ptransp

a*: scales -> pmisc

x*: proportioners, feeders, distribution facilities-> pmisc

m*,y*: motors, engines, drives-> pmachine

x[1]: miscellaneous-> pmisc

Symbols to be supported are described in [S7].

B.3.4.2 Single Line Symbols

First character for single line symbols is:

- S/s: Single line

Structuring/naming conventions (2nd character)to subdirectory

b: sensors -> pinstrum

m: motors -> pinstrum

n: controllers -> pinstrum

p: indicating instruments-> pinstrum

y: actuators -> pinstrum.

---

[1] Not defined within ISO.

B.3.4.3 Miscellaneous

Structuring/naming conventions (1st character)to subdirectory

q: qualifying symbols-> pqualif.

B.3.5 Drawing Application

Subdirectories, including symbols for drawings, are all located below directory ${C_PROJ_LIB_DIB}/drs*<standard><language>*/blksld.

B.3.5.1 Graphic Symbols

Structuring/naming conventions (1st character)to subdirectory

G/g: pure graphical symbols (first character).

B.3.5.2 Single Line Symbols (for Overview Diagrams)

Symbols to be available are described in [S2].

First character **s**:

Structuring/naming conventions (2-3 character)to subdirectory

ag: groups, composed symbols-> ogroups

b*: sensors -> oinstrum

c*: capacitors -> opasscom

f*: protective devices-> oprotect

h*: signalling instruments-> oinstrum

m*: motors -> opower

n*: amplifier/controllers-> oinstrum

p*: indicating instruments-> oinstrum

q*: power circuit devices-> opower

r*: resistors -> opasscom

t*: transformers -> opower

u*: transducers -> oinstrum.

B.3.5.3 Block Symbols (for Overview Diagrams)

First character is **b**

Structuring/naming conventions (2nd character)to subdirectory

(not yet specified)-> oblock.

B.3.5.4 Miscellaneous Symbols (for Overview Diagrams)

Structuring/naming conventions (1st character)to subdirectory (net q: Qualifying symbols-> oqualif

NOTE

See qualifying symbols for circuit diagrams; for special qualifying symbols needed for instrumentation see [D5].

p: P&I diagram symbols used in single line-> oprocess.

Appendix C Guideline for Building Symbols

C.1 Introduction

C.1.1 Background

In symbol blocks, you can specify the following:

- Which static graphic symbols are included
- Which attribute definitions are related to the symbol
- Which connect nodes/connection points are related to the symbol.

The graphical representation is separated between different orientation views/variants corresponding to the rotations, in which the symbols can be used within circuit diagrams. If this is the same, symbols must be specified several times.

C.1.2 General Conventions

Symbols are stored on the file system as AutoCad files in .dwg file format.

Conventions concerning use of layers for circuit diagram and layout diagram symbols are described in the Electrical Diagram Builder reference manual.

C.2 Circuit diagram symbols

C.2.1 Reference to standards

The graphic part of conventional circuit diagram symbols according to IEC and DIN must be done according to [I2]. More complex symbols may be composed also by several elemental symbols.

Symbols specific to instrumentation and control according to DIN are also described in [D5].

Symbols specific to instrumentation and control according to ISO are also described in [S2].

Principles and rules according to ISO/IEC are described in [S5].

Principles and rules according to DIN are described in [D6] and [D2] T1000.

C.2.2 Graphical design principles

General graphical conventions:

- Grid system (Module) for format size <= A3
  M=2.5 mm for IEC symbols
  M=2 mm for DIN symbols

- Minimum distance between graphic lines or graphic lines and text = 0.2 x M
  - 0.5 mm for IEC
  - 0.4 mm for DIN
- Minimum height (h) for attributes = 1xM
  - IEC: 2.5 mm
  - DIN: 2.0 mm
- Minimum distance between text/attributes lines >= 1.3xh
  Recommended values are:
  - IEC symbols: 3.5 mm
  - DIN symbols: 3 mm
- Properties of graphics in symbols:
  - Thickness: (d) is 0.1xM
  - Layer: 152
  - color: by layer (white)
- It is recommended that you block graphical symbols before adding attributes.

C.2.3 Attribute conventions.

Item Designation Attributes

Item Designations in this context means designation of objects but without redundant parts (which may be defined already on sheet or boundary frame level).

ID: Item Designation
Is used for the identifying product item designation, if the product is identified in location oriented way.
Otherwise it's used only for identification of the location of the item.

IDF: Function oriented Item Designation (=)
Is used for the identifying product item designation, if the product is identified in product or function oriented way.

Recommended attribute properties are:
- Style: ITEMDES
- Layer: 9

Device channel related attributes

Daughter symbols representing device channels can also get the attributes

CHID: Channel Id (optionally)
Item designation of channel on multi channel boards

Recommended attribute properties are:
- style: STANDARD
- layer: $A_REM

DCS/PLC related attributes

Symbols representing DCS/PLC related objects (for example, signal DB element) also get the attributes:

SYM_ADR: Symbolic Address/Name
Symbolic identifying name of elements within DCS/PLC system
(for example, signal name)

ABS_ADR Abs(olute) Address (optionally)
internal identifier of elements within DSC/PLC system, describes for example full address like
<net>.<node>.<bus>.<station>.<board>.<channel>
(for example, "13.10.AI3.5")

PLC_TYP: Type of element (within DCS/PLC system) (optionally)

POS: Position of DCS/PLC board within rack (optionally)

Recommended attribute properties are:
- style: STANDARD
- layer: $A_REM

Terminal/Connection point related attributes

For symbols representing terminal block (channel)s this attribute is relevant:

***A:** Terminal number (on terminal bar)

For symbols having connection points the connection point numbers/id's are handled in attributes according to the naming coventions:

na: Terminal number
Examples: 1A, 2A, 3I, 4I, 5E, 6E,....
The number n must be unique for one symbol
For further description see Section C.2.4, Connection Point and Reference Point Definitions

Recommended attribute properties are:
- style: TERMINALS
- layer: 8

Article library related and descriptive attributes

LIB:	Library Identifier
TY:	Type designation
TPN:	Type number
FAB:	Manufacture
ELD:	Electrical data
DIM:	Dimension
FRD:	Article Number
PRI:	Article Price
NA:	Group Code
EG1:	Remark 1

Recommended attribute properties are:
- style: STANDARD
- layer: $APP (normally not visible)

EG2: Remark 2

Recommended attribute properties are:
- style: STANDARD
- layer: $A_REM (normally visible)

free attributes: Recommendation: tag name of attribute should be identical to attribute name within data base object.

Attributes for Cross-Reference handling

CREF: Cross-reference value
(Remark: already defined in elder symbols)
Used for daughter and cross-reference symbols
Recommended attribute properties are:
- style: CREF
- layer: $CREF

CRTYP: Cross-reference type
Describes type of cross-reference for daughter and cross-reference symbols
Example values for cross-reference symbols: RSC, RSO, RL2
Example values for daughter symbols: KSC, KLO, KL2, KS2
Attribute properties are:
- style: STANDARD
- layer: $A_REM (normally visible)
- must be set to invisible and constant

Attributes specific for application symbols:

BLAD: Reference sheet identifier/number
Used for signal/connection line referencing symbols
Recommended attribute properties are:
- style: STANDARD
- layer: 8

ELP: Electrical Potential /Signal Item Designation
Used for signal/connection line referencing symbols
Recommended attribute properties are:
- style: STANDARD
- layer: 8

VAR, VAL Variable name and Variable Value
Used for some application specific symbols to assign certain information to connect nodes or connection lines.

C.2.4 Connection Point and Reference Point Definitions

An electrical connection point is a logical connection point of a symbol, to which only electrical conductors (by use of connection lines) can be connected.

The reference point corresponds to that point, which is used later to insert the symbol into the diagram. Therefore, it is often also called the insertion point.

Each electrical connection point defined for a symbol block consists of two parts:

- The attribute related to the symbol block to handle the connection point/terminal designation/number. The TAG name of the attribute has to follow the naming conventions 'na' (a=(A,B,...Z); n=(1,2,3,....).
  The number 'n' must be unique for all connection points of a symbols
- A "connect node" (which consists of a block named NOD) representing the connection point itself, to which connection lines can be assigned or which break up connection lines automatically. The "node" name has to follow the naming conventions 'an'.

Recommended letter for connect nodes to "internal" side (up to DCS system) is "I"
(examples: I1, I2, I3)
Recommended letter for connect nodes to "external" side (down to the field) is "E"
(examples: E1, E2, E3).
Recommended letter for connect nodes for other use (for example, power supply) is "A"
(examples: A1, A2, A3)

Connect nodes must have a distance of n x 2M

For numbering of connection points you have to follow the rule according to [D6].

- The reference point is defined for a symbol in basic orientation and remains independent from its rotation
- For symbols with vertical basic orientation the reference point is on upper side the most left connect node
- For symbols with horizontal basic orientation the reference point is on left hand side the most upper connect node
- The connect node identical to the reference point is the first one
- The other connect nodes are numbered sequentially in clockwise sense

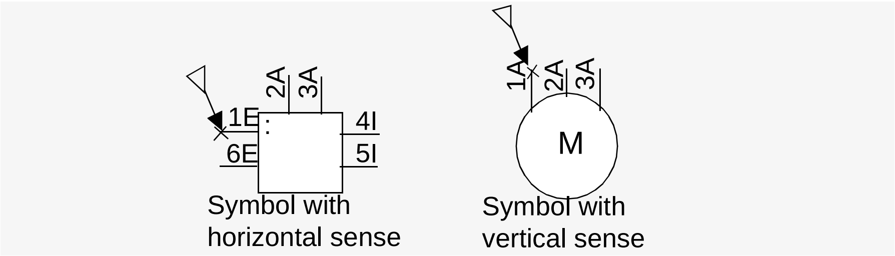


Figure C-1. Reference points and connection point numbering for symbols

C.2.5 Symbol Groups/Classes

The Symbol Blocks represent AdvaBuild database objects in the diagrams. Symbol blocks within circuit diagrams are used therefore for the following purposes:

- Mother symbols
  Note: Devices without channels will also be represented by mother symbols
  - for general devices
  - for PLC/DCS boards.
- Daughter symbols
  - for general device channels with referencing by channel (conventional and instrumentation)
  - for general device channels with referencing by connect nodes
  - for PLC/DCS board channels.
- Cross-reference Daughter symbols
  - for general device channels with referencing by channel (conventional and instrumentation)
  - for general device channels with referencing by connect nodes
  - for PLC/DCS board channels.
- Terminal Symbols
  - for “single channel” terminal blocks
  - for “multi channel” terminal blocks.
- Signal reference Symbols
  - to potential
  - within sheet
  - between sheets.

C.2.5.1 Mother symbols for devices

Mother symbols represent devices, which are handled within the AdvaBuild database by objects of type DEV respectively DCS_DEV. The mother symbol represents the device including its power supply within a circuit diagram.

Necessary attributes:

- ID: Item Designation
- IDF: Func. Item Des.

Optionally the article library related and descriptive attributes, as well as connection point definitions, can be used:

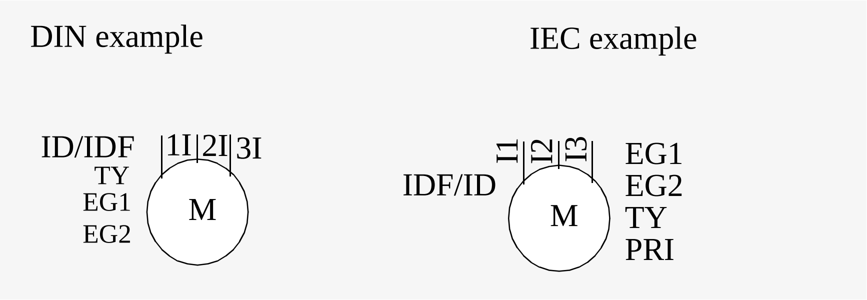


Figure C-2. Examples for mother symbols

Mother symbols for PLC/DCS devices represent PLC/DCS boards, which are handled within the CAPE database by objects of type DCS_DEV.

Optionally those symbols get the additional attributes:

- ABS_ADR: DCS system internal board address
- SYM_ADR: DCS system symbolic board name.

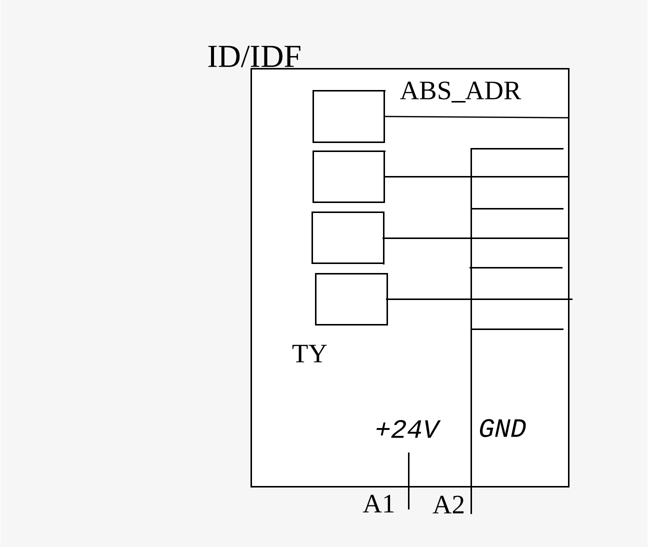


Figure C-3. Examples for mother symbol for DCS Device

C.2.5.2 Daughter symbols for device channels

If a symbol block represents a device channel, it is called "daughter symbol".

Necessary are the attributes:

- ID: Item Designation
- IDF: Func. Item Des.
- CREF: Ref. Designation
- CRTYP: Ref. Type (Set invisible).

For daughter symbols identified by connect nodes (for example, relay contact symbols) the connect node definitions are also necessary.

Necessary attributes for daughter symbols identified by channel number:

- CHID: Channel Item. Des.

Optional attributes for daughter symbols representing DCS I/O channels:

- SYM_ADR: symbolic address within control system, here Signal Name
- PLC_TYP: element/object type within control system, here Signal type
- ABS_ADR: System Item. Des. (for example, "13.10.AI3.5").

Optionally the following can also be included:

- descriptive attributes
- connection point definitions.

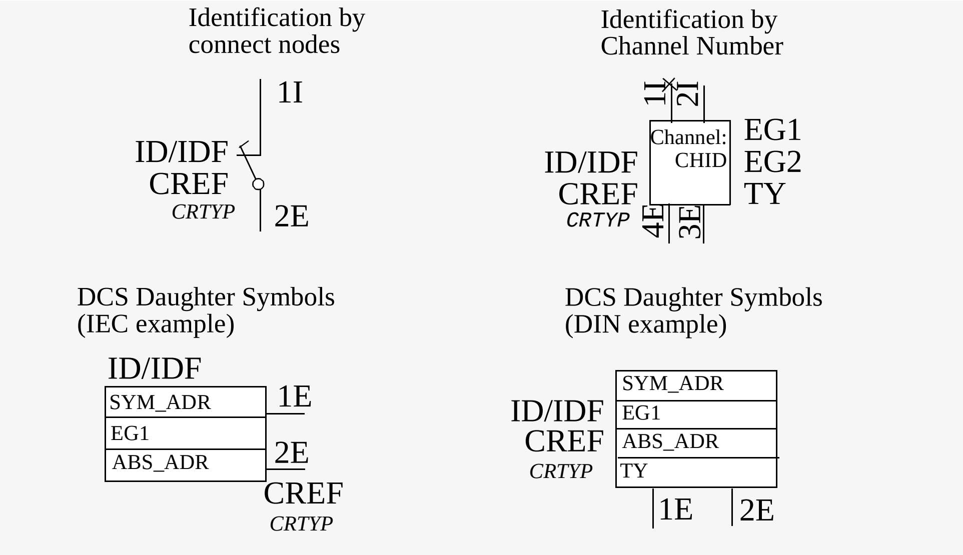


Figure C-4. Examples for daughter symbols

C.2.5.3 Cross-reference symbols for device channels

For cross-reference handling for each used channel of a multi channel device must be placed a cross-reference symbol assigned to the mother symbol as counter part.

Important hint:

The assignment can be done in two different ways:

1. Mother and cross-reference symbols can be grouped by a boundary frame (should be preferred for I/O boards).
2. Mother and cross-reference symbols can be assigned by a mechanical link (should be preferred for relays).

A "Cross-reference symbol block" should include only the necessary attributes:

- ID: Item Designation
- CREF: Ref. Designation
- CRTYP: Ref.Type (set invisible).

For cross-reference symbols identified by connection points (for example, relay contact symbols) the connection point definitions are also necessary.

Necessary attributes for daughter symbols identified by channel number and DCS I/O channels:

- CHID: Channel ID.

Optionally connect node definitions can be included.

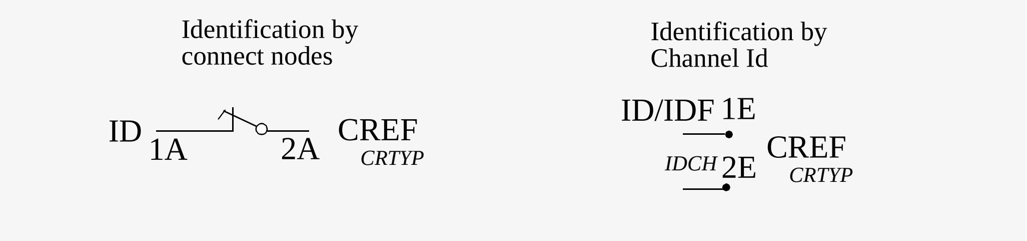


Figure C-5. Examples for cross-reference symbol

NOTE

The "cross-reference symbol blocks" may optionally already be grouped together with the mother symbol block definition (so the referenced mother symbol is originally stored already as composed symbol block).

C.2.5.4 Terminal symbols

Definitions:

Terminal or better **terminal block** means physically a component which can be used to interconnect cable cores. Terminal (blocks) are normally physically snapped on a terminal bar.

Often terminal (blocks) provide several "channels" isolated to each other (for example, two-tier terminal, three-conductor terminal). In this case the terminal block is called "multi channel terminal block".

A **Terminal symbol** represents within a circuit diagram only one "terminal channel" of a terminal (block).

Necessary for a terminal symbol are the following attributes:

- ID: Item Designation (Identifying designation of BAR object)
- *A: Terminal No.
- two connect node definitions (for example, A1, A2) by use of **NOD** symbol

Recommended to be used are the attributes:

- IDF: Func. Item Des.

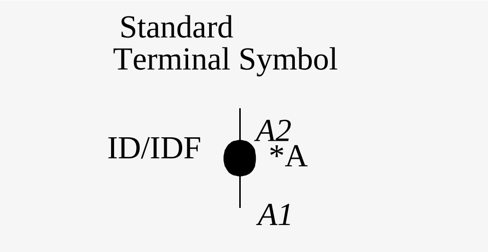


Figure C-6. Examples for terminal symbols

C.3 P&I Diagram/Drawing symbols

C.3.1 Reference to standards

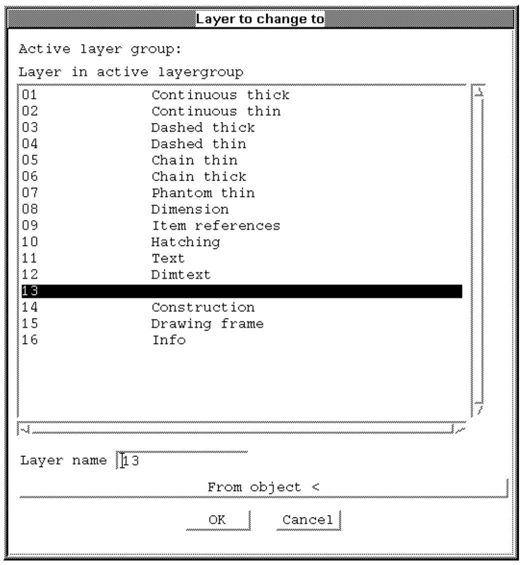
Relevant symbols and drawing rules according to DIN are described in [D7]. Symbols specific to instrumentation and control are also described in [D5].

Relevant symbols and drawing rules according to ISO are described in [S6]. Symbols specific to instrumentation and control are also described in [S2].

Symbols relevant according to ISO for fluid power systems are described in [S10].

C.3.2 Graphical design principles

- Grid system (Module)
  M=2.5 mm for IEC and DIN symbols
- Properties of graphics:
  - Thickness: (d) is 0.1xM (corresponds to layer 02 of related layergroup).



```
Active layer group:
Layer in active layergroup
01              Continuous thick
02              Continuous thin
03              Dashed thick
04              Dashed thin
05              Chain thin
06              Chain thick
07              Phantom thin
08              Dimension
09              Item references
10              Hatching
11              Text
12              Dimtext
13
14              Construction
15              Drawing frame
16              Info

Layer name 13
```

Figure C-7. Layer to change to

- It is recommended that you block graphical symbols before adding attributes
- Minimum distance between graphic lines or graphic lines and text:
  - 0.5 mm for A4..A3
  - 0.7 mm for A2..A0
- Minimum distance between text lines/attributes (Corporate Standard) >= 1.3xh.

Attributes for ISO/IEC and DIN symbols

- Item designations
  - Height 2.5 mm Line distance 3.5 mm for format A4, A3 (only for drawings)
  - Height 5 mm, Line distance 7 mm for format A2..A0 (for drawings and P&I diagrams)
- Other text/attributes (article attributes, remarks)
  - Height 2.5 mm, distance between lines 3.5 mm.

C.3.3 Attribute conventions

Necessary are the attributes:

- ID: Item Designation
  Remark: should be set invisible, if tagname is written in split way.

Optionally for item designations the following can also be used:

- ID1, ID2 for split writing of the tagname (for example, TICA 100).

Attribute properties:
- Style: STANDARD
- Layer: layer 09 of related layer group
- Adjust: depends on position (preferred: Left, Center, Middle).

For optional article related and descriptive attributes see definitions for circuit diagram symbols. Recommended attribute properties are:
- Style: STANDARD
- Layer: layer 11 of related layer group.

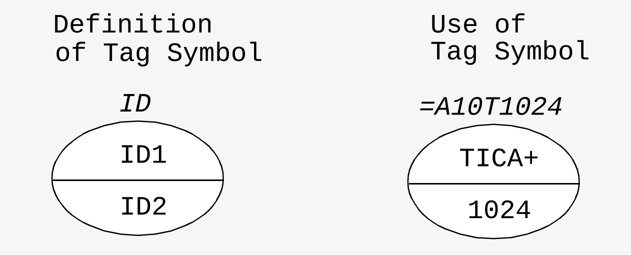


Figure C-8. Example for a tag symbol

C.4 Layout symbols

C.4.1 Graphical design principles

- Symbols are drawn with 1:1 scaling
- For use of layers for graphics see reference manual *Electrical Diagram Builder Layout.*

C.4.2 Attribute conventions

Layout symbols (only relevant for layout diagrams) represent the physical layout, for example, of devices, cabinets or racks.

Necessary are the attributes:

- **ID**: Item Designation
  Attribute properties:
  - Style: STANDARD
  - Layer: layer 11 of related layer group (A11, P11, C11)
  - Adjust: depends on position (preferred: Left, Center, Middle)
  - Height: no fixed recommendation, must be in good relation to size of symbol. Preferred height is 25 mm.

Optionally article library and descriptive attributes can also be used:

- **EG1**: Remark1
- **EG2:** Remark2
- **TY**: Type
- **NA**: Group Code
- **BEN**: Name
- **FAB:** Manufacturer
- **TPN:** Article Number
- **ELD**: Electrical Data
- **DIM:** Dimension
- **FRD**: Article Number.

Attribute properties are:
- style: STANDARD
- preferred text height: 3,5 mm
- layer: D11 (design aid group)
- adjust: left
- angle: 0 degree

Optionally the following can also be used:

- **PUNCH**: reference to a punch symbol to be inserted automatically
  Remark: referenced punch symbol is inserted on same position as insertion points
  Hole punch reference attributes PUNCH,...
  - style: STANDARD
  - text height: 3,5 mm
  - layer: H11
  - adjust: left
  - angle: 0 degree

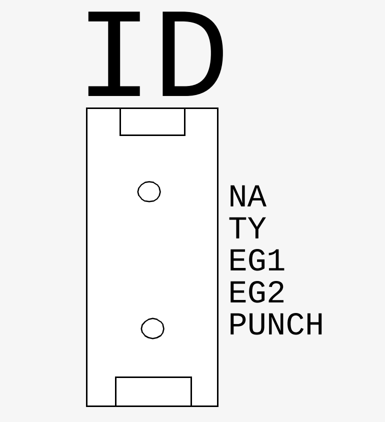


Figure C-9. Example for layout symbol

C.5 Hookup symbols

C.5.1 Graphical design principles

Hookup diagrams normally describe the interconnection and assembly of sensor or actuator related equipment and accessories.

There are different possible types of hookups. One type is made up more or less like a single line diagram and describes the interconnections only in a logical way. The other type is made up more like a mechanical drawing and shows the exact layout of the equipment and its arrangement. Therefore there are no fixed recommendations on how to make symbols for hookup diagrams.

If the real layout is shown in hookups, mechanical symbols should be drawn with 1:1 scaling.

C.5.2 Attribute conventions

Necessary are the attributes:

- REFNO: Reference number
  Recommended attribute properties:
  - Style: STANDARD
  - Layer: layer 09 of related layer group
  - Adjust: depends on position (preferred: Left, Center, Middle).

Optional attributes are the article library related and descriptive attributes described for layout symbols. Recommended attribute properties are:
- Style: STANDARD
- Layer: layer 11 of related layer group.

Appendix D Internal Variables and Environment Variables

D.1 Introduction

Using internal variables and environment variables lets you configure how the (Electrical) Diagram Builders operate when started and when printing. You can specify various select statements within file dib.sql, the setting of own or preset internal and environment variables within files <object type>.dib, and the title block mapping file dib.dmi.

D.2 Syntax Definition

Use these constructions to describe the syntax for using internal variables and environment:

- [*expression*] -- The *expression* is optional
- {*expression*} -- The *expression* may occur many times or never.
- *expression_1* | *expression_2* -- either *expression_1* or *expression_2* can occur
- ‘*text*’ -- the text described by text must occur
- ***NAME*** -- An expression that is defined by ***NAME*** can occur
- ***NAME*** = *expression* -- An *expression* will be defined and assigned to ***NAME***.

Table D-1. Specification of variable expressions

ITEM		DEFINITION
EXPRESSION	=	[TEXT] {VAR_EXP [TEXT]}
VAR_EXP	=	ENV_VAR_EXP | INT_VAR_EXP
ENV_VAR_EXP	=	‘${‘ ENV_VAR_NAME [OPTION] ‘}’
INT_VAR_EXP	=	‘@{‘ INT_VAR_NAME [OPTION] ‘}’
ENV_VAR_NAME	=	EXPRESSION -- The result of evaluating the expression must be a regular UNIX environment name.
INT_VAR_NAME	=	EXPRESSION -- The result of evaluating the expression must be an IDENTIFIER (see below)
OPTION	=	‘+’ TEXT [‘:-’ TEXT] | ‘:-’ TEXT [‘:-’ TEXT]
TEXT	=	PRINT_CHAR {PRINT_CHAR}
IDENTIFIER	=	IDENTFIER_CHAR {IDENTIFIER_CHAR}
IDENTFIER_CHAR	=	ALPHA | NUM | ‘_’ | ‘.’ | ‘(‘ | ‘)’ | ‘[‘ | ‘]’
ALPHA	=	‘A’ - ‘Z’ | ‘a’ - ‘z’
NUM	=	‘0’ - ‘9’
PRINT_CHAR	=	any printable character as defined in isprint character set table (see also man 3c isprint) excepting character combinations ‘${’, ‘@{’, and ‘}’.

According to the given syntax specification the following expressions are valid:

- This_is_a_simple_expression
- This_has_one_@{int_var}
- This_has_@{two}_@{int_vars}_and_stuff
- This_has_@(a_@{nested}_variable}
- This_has_@{an:+option}_if_defined
- This_has_@{an:-option}_if_not_defined
- This_has_@{both:+plus:-minus}_options.

You may use also environment variables like this or mix it.

Table D-2. Specification of variable definition

NAME		DEFINITION
VAR_DEF	=	INT_VAR_DEF | ENV_VAR_DEF
INT_VAR_DEF	=	‘@’ INT_VAR_NAME [‘=’ [START EXPRESSION STOP]]
ENV_VAR_DEF	=	ENV_VAR_NAME [‘=’ [START EXPRESSION STOP]]
START	=	any character but not ‘ ‘ (blank)
STOP	=	the same character as defined by START

According to the given syntax the following definitions are valid:

- @int_var = ‘an ordinary text’
- @int_var = “a text containing a ‘literal’”
- @int_var = #a text containing a “string” and a ‘literal’#
- @int_var
  removes the internal variable
- @int_var=
  removes the internal variable
- @int_var= ““
  assigns an empty string
- @env[ZORRO] = ‘${ZORRO:-}’
  From now you can use @{env[ZORRO]} or ${ZORRO}.
- @HomeOf(${LOGNAME})=’${HOME}’
  If LOGNAME was set to master the internal variable master is set as
  @{HomeOf(master)} is set to ‘/users/master’.

D.3 Semantic Definition

Assignments to internal variables and environment variables have the following semantics:

- *@name* = '*expression* '
  assigns the value which results from expanding the *expression* to the internal variable *name*. If the variable *name* already exists it will be overwritten. If the variable *name* does not exist it will be created.
- *name* = '*expression* '
  assigns the value which results from expanding the *expression* to the environment variable *name*. If the variable *name* already exists it will be overwritten. If the variable *name* does not exist it will be created. The environment variable will also be exported to all subprocesses (for example, AutoCAD).
- *@name* = or *@name*
  deletes the internal variable with name *name* if it exists. If the variable does not exist nothing will happen.
- *name* = or *name*
  is identical to *name*=''. No environment variable can be deleted.

Option handling within internal variables and environment variables has the following semantic:

- @{*name*:-*text*}
  If the internal variable *name* is undefined the given *text* will be taken; otherwise the value of variable *name* will be taken.
- @{*name*:+*text*}
  If the internal variable *name* is defined the given *text* will be taken; otherwise the value of variable *name* will be taken.
- @{name:-text1:+text2}
  If the internal variable *name* is defined the given *text1* will be taken; otherwise the given *text2* is taken.
- ${*name*:-*text*}
  If the environment variable *name* is undefined the given *text* will be taken; otherwise the value of variable *name* will be taken.
- @{*name*:+*text*}
  If the environment variable *name* is defined the given *text* will be taken; otherwise the value of variable *name* will be taken.
- @{name:-text1:+text2}
  If the environment variable *name* is defined the given *text1* will be taken; otherwise the given *text2* is taken.

D.4 Complex Examples

Internal variables are normally used only to define certain configuration information (as shown in ci_dia.dib, for example) or are used to reference values selected from the data base (as shown in dib.sql and dib.dmi). However, you may use internal variables and environment variables to define various mechanisms such as mappings, to compare values, or to check for empty strings.

D.4.1 Checking for an Empty String

NOTE

The variable @IsEmptyString is already predefined and is used within dib.dmi.

The following definition must be done before checking for an empty string:

– @IsEmptyStr="IsEmptyStr"

Now the following expression checks if the variable @string is empty and expands to the text 'is empty' relative to 'is not empty'.

– @{IsEmptyStr@{string}:+is empty:-is not empty}

If @string is empty the expression will be first expanded as

– @{IsEmptyStr:+is empty:-is not empty}

As @IsEmptyStr is a defined variable the positive option will be chosen.

If @string is set to 'text' the expression will be first expanded as

– @{IsEmptyStrtext:+is empty:-is not empty}

As @IsEmptyStrtext is an undefined variable the negative option will be chosen.

The example from within dib.dmi expands either PROJECT.OWNER or DSTRUC.OWNER. This is dependent on whether the value of DSTRUC.OWNER is an empty string or not.

OWNER
@{@{IsEmptyStr@{DSTRUC.OWNER:-}:+PROJECT.OWNER:-DSTRUC.OWNER}:-}

D.4.2 Defining Mappings

You may want to define certain mappings of a value stored within an environment variable or internal variable:

– ValueOf(name1)='value1'

– ValueOf(name2)='value2'

– @Value = 'ValueOf(@{name))'

In this example the variable @name is preset and is used to detect the right value from the mapping @ValueOf.

D.4.3 Comparing Values

Use the following construction to test whether the values of two variables @one and @two are identical:

- @.-.@{one}=’’
- @.-.@{two}=
- @{.-.@[one}:-values are identical:+values are different}

The variable construction @.-..name is used to guarantee that the names were not used before:

1. If @one=’left’ and @two=’right’ then
    a. @.-.@{one}=’’ ==> @.-.left=””
    A variable with name .-.left will be created and is set to an empty string.

    b. @.-.@{two}= ==> @.-.right=
    A variable .-.right will be deleted if existent. As this variable does not exist, nothing will be done.

    c. @{.-.@[one}:-values are identical:+values are different} ==>
    @{.-.left:-values are identical:+values are different} ==>
    values are different

    Since the contents of @one (that is, ‘left’) and @two (that is, ‘right’) are different, the variable @.-.@{one} (that is, @.-.left) will not be deleted by the expression @.-.@{two}= (that is, @.-.right=).

2. If @one=’equal’ and @two=’equal’ then
    a. @.-.@{one}=’’ ==> @.-.equal=””
    A variable with name .-.equal will be created.

    b. @.-.@{two}= ==> @.-.equal=
    The variable with name .-.equal will be deleted.

    c. @{.-.@[one}:-values are identical:+values are different} ==>
    @{.-.equal:-values are identical:+values are different} ==>
    values are identical - as @.-.equal does not exist.

    Since the contents of @one and @two are identical, the variable @.-.@{one}
    (that is, @.-.one) will be deleted by the expression @.-.@{two}= (that is, @.-.one=).

NOTE

This solution helps only if the expressions @{one} and @{two} expand to regular identifiers for internal variables. (@one=”name=’zorro’” would cause problems.)

INDEX

C

Configuring plotter 3-38

D

Diagram object attributes
  DOC_TYPE 4-24
  Edit with Template Builder 4-24
  LANGUAGE 4-24
  LAYOUT_FORMAT 4-24
Diagram objects
  Copy 4-26
  Copy with Template Builder 4-26
  Delete 4-25
  Open graphical content 4-25
  Rename 4-26
Diagram view 4-25
Diagrams
  Edit graphics 4-1
  Print directly 4-27
  Print more than one 4-28
Directories
  ${C_CAPE}//usr/. 4-27
Directory structure levels
  3-1
  Global level 3-1
  Project 3-1
  System 3-1
  User 3-1
DOS 3-37
dwg 5-1
dwg files 3-37

E

Export/Import facility 5-1

F

Font files 3-37

G

Global 3-1 to 3-2
Global level directories
  C_CAPE_ALL_ACAD 3-1 to 3-2

I

Importing drawings done on PC 5-2
Insert diagram objects 4-21

L

LANGUAGE 4-24

O

Object types
  DRAWING 4-1
Operating parameters
  Alarm on Error
    3-37
  Automatic Audit after IGESIN, DXFIN, or DXBIN 3-38
  Default Plot File Name 3-37
  File Locking 3-38
  File name case 3-37
  Full-Time CRC Validation 3-38
  License Server 3-38
  Login Name 3-38
  Network Node Name 3-37 to 3-38
  Placement of Temporary Files 3-37
  Plot Spooler Directory 3-37

P

Postscript files 4-27 to 4-28
Printer queue 4-27
  ps 4-28

S

Structure Builder 4-2, 4-21, 4-25 to 4-27

T

Template Builder 4-2, 4-21, 4-25
Template view 4-2

U

UNIX 5-2

V

Variant Handling 3-16

3BSE 001 979R0201

February, 1997